CAMPUS
DE EXCELENCIA
INTERNACIONAL

Graduado en Ingeniería Informática

Universidad Politécnica de Madrid

Facultad de Informática

TRABAJO FIN DE GRADO

Diseño e implementación de una herramienta para el reconocimiento automático de términos compuestos dentro de vocabularios biomédicos

Autor: Antonio Rico Díez

Director: Raúl Alonso Calvo

MADRID, MAYO DE 2013

Abstract

In the latest and most used biomedical languages, we usually find term composition operations from existing terms. These mechanisms increase the utility of those terminologies they belong to. Despite this, these operations present a disadvantage, that is, the possibility of representing the same concept with different base concepts which introduces a certain degree of ambiguity in those complex terms.
The objective of this final degree project consists in developing a tool that allows recognizing terms from those complex biomedical vocabularies, that is, terminologies with terms comprised of simpler terms such as SNOMED.
By completing this project, we obtained a tool capable of identifying the present ambiguities in the representation of those composite concepts and represent them in a homogenous format.
To facilitate the interoperability and accessibility of the tool it was decided to offer it through a web interface loadable from any place or device with access to the internet.

Resumen

En los vocabularios biomédicos actuales más utilizados, suelen existir mecanismos de composición de términos a partir de términos pre-existentes. Estos mecanismos de composición aumentan la potencia de los lenguajes que los poseen pero parten con la desventaja de la posibilidad de representar un mismo concepto con diferentes conceptos base, lo que incluye un componente de ambigüedad en los mismos.
Este trabajo de fin de grado consiste en la realización de una herramienta que permita reconocer términos de estos vocabularios biomédicos complejos, es decir, vocabularios con términos compuestos por otros términos como puede ser el caso de SNOMED.
Con la consecución de este proyecto, obtendremos una herramienta capaz de identificar las ambigüedades presentes en la representación de estos conceptos compuestos y representar de una forma homogénea dichos conceptos.
Para favorecer la interoperabilidad y accesibilidad de la herramienta se ha decidido ofrecerla mediante una interfaz web accesible desde cualquier dispositivo o lugar con acceso a internet.

Agradecimientos

En la realización de este proyecto he podido contar con la ayuda y apoyo de una gran cantidad de personas. Quisiera dar las gracias especialmente a mi tutor, Raúl Alonso Calvo y a David Perez del Rey quienes me han ayudado y aconsejado durante toda la duración de este proyecto. También dar las gracias a mis compañeros de trabajo, Sergio, Juanma, Santi y Gema, con quienes he compartido la experiencia de trabajar en la temática de la interoperabilidad semántica y que me han proporcionado una gran ayuda.También dar las gracias a mi familia cuyo apoyo constante me ayudo a completar de forma satisfactoria este proyecto y me ha permitido continuar con entusiasmo toda la carrera. Finalmente, dar las gracias también a todos los participantes en los proyectos INTEGRATE y EURECA en cuyo marco está enfocado este proyecto.

Índice general

Capítulo 1

INTRODUCCIÓN

1.1 Planteamiento del problema

SNOMED-CT, LOINC, HGNC, MedDRA,... Estos son solo algunos ejemplos de vocabularios biomédicos que se pueden encontrar hoy en día. Muchos de estos vocabularios se especializan en un ámbito específico de información médica, por ejemplo, HGNC se centra en la representación de los genes, LOINC en información de observaciones de laboratorio...

Como podemos imaginar, para poder representar toda la información disponible en ocasiones nos vemos obligados a utilizar conceptos de diferentes terminologías. Algunas de estas terminologías también han implementado mecanismos de composición de conceptos, esto es, la utilización de los términos existentes para crear otros nuevos que contengan la información que necesitamos. Por ejemplo, fractura en el fémur de la pierna derecha podría ser representado como:

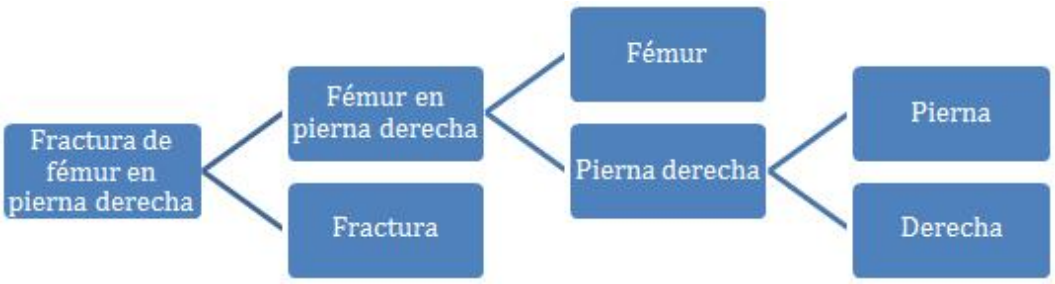

Figura 1.1: Descomposición de término compuesto

Este tipo de composición incrementa exponencialmente la capacidad de representación de información del vocabulario al permitir la composición de conceptos complejos a partir de otros más simples. También aumenta la capacidad de análisis sobre esos conceptos al estar formados por conceptos básicos, es decir, es más fácil entender los conceptos simples de los que están compuestos los términos compuestos que el término compuesto en si mismo. Con esto podemos afirmar que los vocabularios médicos más completos deben poseer esta herramienta como mecanismo de ampliación del vocabulario.

El problema tratado en este proyecto es el de reconocer estos términos compuestos de una forma comprensible y efectiva para el usuario para así poder descomponerlos o bien averiguar en qué términos compuestos puede estar presente otro término (compuesto o simple).

Para realizar este proyecto nos centraremos en la terminología SNOMED-CT [1][2] la cual incorpora los previamente mencionados mecanismos de composición además de ser una de las terminologías con una mayor colección de conceptos representados con más de 311.000 términos y más de 1.300.000 relaciones. Estas relaciones pueden indicar relaciones de subclase o descendencia (relaciones "is a") u otra serie de relaciones que indican la composición del término. Por ejemplo:

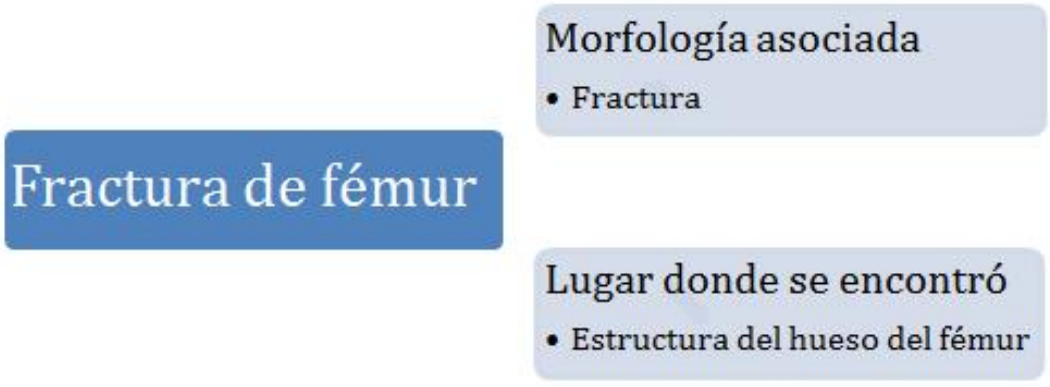

Figura 1.2: Relaciones del concepto Fractura de fémur

En este ejemplo, "morfología asociada" y "lugar donde se encontró' serían las relaciones mientras que "Fractura" y "Estructura del hueso del fémur" serían los valores de dichas relaciones. SNOMED-CT incorpora una representación que incorpora los elementos básicos de los que está compuesto un término, la forma normal. Esta forma normal [3] será uno de los principales puntos de estudio de este proyecto.

Cabe destacar que SNOMED-CT y los mecanismos de composición de este lenguaje son uno de los pilares utilizados en la obtención de interoperabilidad semántica en los proyectos europeos INTEGRATE [4][5] y EURECA [6] los cuales se introducirán más adelante. La interoperabilidad semántica se refiere a la posibilidad de que varios sistemas de información sean capaces de entenderse. Esto se puede lograr mediante el establecimiento de un modelo de datos común y una codificación común en los conceptos médicos. Para esto es especialmente útil la descomposición de los términos a una serie de conceptos básicos comunes lo cual es facilitado mediante el análisis de los términos compuestos de la terminología SNOMED-CT.

1.2 Objetivos

Como hemos mencionado en la sección anterior, el reconocimiento de los términos compuestos es uno de los desafíos presentes en las terminologías médicas actuales y es un mecanismo utilizado en diversos proyectos europeos como por ejemplo: INTEGRATE y EURECA.

Los objetivos marcados para la realización de este proyecto se centran en crear una herramienta que permita obtener y estudiar la estructura de los conceptos de la terminología SNOMED-CT. Tras un proceso de estudio para maximizar el alcance de la herramienta a la mayor parte de la comunidad médica se decidió realizarla como una aplicación web para permitir el acceso desde cualquier lugar y desde cualquier dispositivo (siempre que se usen las últimas tecnologías).

- **La herramienta debe permitir la búsqueda de conceptos de la terminología SNOMED.**
  La herramienta presentará un buscador de conceptos que permita búsqueda mediante el nombre del concepto o el código dentro de la terminología SNOMED-CT.

- **La herramienta debe realizar un análisis de los términos compuestos de las búsquedas.**
  La herramienta analizará los conceptos buscados determinando si se trata de un concepto compuesto o simple, presentando la forma descompuesta en caso de ser un término complejo.

- **La herramienta debe permitir la obtención de una jerarquía con descendencia y ascendencia de los conceptos.**
  La herramienta presentará de una forma visual la posición de los conceptos dentro de la jerarquía del vocabulario SNOMED-CT.

- **La herramienta debe permitir obtener los términos compuestos de los que un término simple dado forma parte.**
  También se presentará los términos compuestos en los que alguno de sus términos coincida con un término buscado.

- **Permitir la realización de las operaciones anteriores de una forma rápida y sencilla que permita una comparación entre los conceptos resultantes.**
  Uno de los objetivos se centra en la usabilidad de la herramienta de forma que su uso sea útil para los profesionales del sector y la información se presente con un tiempo de espera mínimo para realizar estudios rápidos de la terminología y de los conceptos en la misma.

1.3 Organización del proyecto

A continuación se detallan las secciones que conforman la presente memoria con una breve descripción de cada una de ellas.

- **Introducción:** En esta sección se presenta el proyecto en sí mismo y sus objetivos.

- **Estado de la cuestión:** En esta sección se explican las diferentes terminologías estudiadas y los mecanismos de composición de términos que contienen además de los proyectos europeos en los que se enmarca el presente trabajo.

- **Tecnologías empleadas:** En esta sección presentaremos las tecnologías empleadas para realizar la herramienta incluyendo su interfaz web y los servicios web creados.

- **Diseño de la herramienta:** En esta sección analizaremos los requisitos encontrados para realizar la herramienta y las decisiones de diseño tomadas para su desarrollo.

- **Desarrollo de la herramienta:** En esta sección detallaremos el proceso de desarrollo de la herramienta, incluyendo las fases y algunos detalles del código fuente.

- **Pruebas de la herramienta:** En esta sección se explicarán las pruebas realizadas para asegurar el correcto funcionamiento de la herramienta y la adecuación de la misma a los objetivos previos.

- **Conclusiones y lineas futuras:** En esta sección se explicará la adecuación de la herramienta a los objetivos previos además de posibles ampliaciones o líneas de desarrollo de la herramienta.

Capítulo 2

ESTADO DE LA CUESTIÓN

2.1 Terminologias estudiadas

2.1.1 SNOMED-CT

2.1.1.1 Características

SNOMED-CT (Systematized Nomenclature Of Medicine Clinical Terms) [1][2] es una de las principales terminologías de conceptos médicos en la actualidad creada por el "College of American Pathologists". Cuenta con más de 311.000 términos y más de 1.300.000 relaciones entre dichos términos. La mayoría de dichas relaciones consisten de la relación "is a" la cual representa una relación de subclase entre dos conceptos. Un archivo OWL conteniendo toda la información de SNOMED-CT puede generarse mediante una herramienta gratuita que provee la "International Health Terminology Standards Development Organisation (IHTSDO)" [7].

Conceptos

Cada término viene representado con un código numérico que representa inequívocamente el concepto. Además contiene la siguiente información:

- **Cero o más descripciones del concepto.** Esto permite realizar búsquedas sobre las diferentes formas de representar cada concepto. Estás descripciones pueden ser sinónimos, nombres completamente especificados o término preferido.
  - **Nombre completamente especificado:** El propósito de esta descripción es describir inequívocamente el concepto y clarificar su significado.
  - **Término preferido:** Nombre usado normalmente por los especialistas del sector.

  - **Sinónimos:** Otras formas de representar el concepto.

- **Una o más relaciones de subclase con otros conceptos.** Excepto el concepto "Concept" de SNOMED el cual es la raíz de todos los conceptos. Esto quiere decir que un concepto puede ser hijo de 2 o más conceptos totalmente diferentes.
- **Cero o más conceptos hijos.**

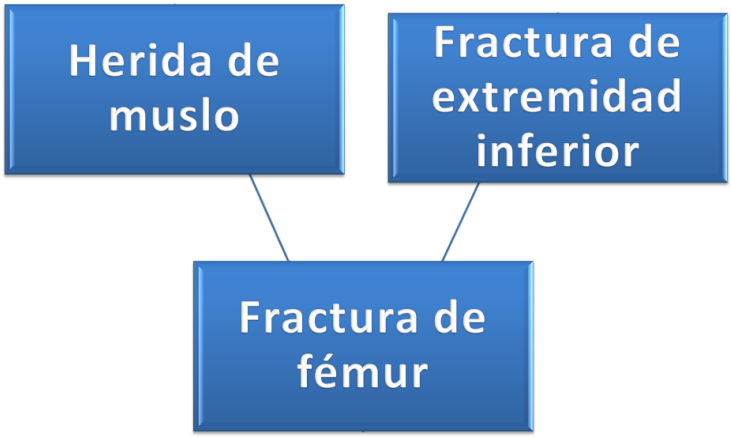

Figura 2.1: Ejemplo de concepto con múltiples padres

Además de esta información también se almacena si el concepto es:

- **Primitivo:** Un concepto es primitivo cuando sus padres y atributos no expresan completamente su significado. Es decir, los conceptos primitivos no tienen las relaciones únicas necesarias para distinguirlos de sus conceptos padres y hermanos.
- **Completamente definido:** El concepto puede ser diferenciado en virtud de sus relaciones. Estas relaciones determinan los términos simples de los que está compuesto el concepto.

Cabe destacar que existen conceptos dentro de SNOMED-CT con una amplia similitud entre ellos. Este es el caso por ejemplo de la rama de Anthracycline cuyo concepto e conceptos hijos presentan dos conceptos con el mismo nombre, uno denotando la sustancia de la que están compuestos y otro denominando el producto en sí.

Relaciones

Existen multitud de relaciones diferentes siendo la relación "is a" un gran porcentaje de ellas. Estas relaciones definen lógicamente el concepto y pueden ser de diferentes tipos:

- **Definitorias:** Estas relaciones son las que definen el concepto. Esto incluye las relaciones "is a" y las relaciones de atributos entre conceptos.
- **Calificadoras:** Estas relaciones se utilizan para calificar conceptos de SNOMED. Por ejemplo, severidad o periodicidad son relaciones que sirven para especificar otro concepto como cáncer o fractura de fémur.
- **Históricas:** Son relaciones que estaban presentes en versiones anteriores pero se conservan por razones de compatibilidad.
- **Adicionales:** Relaciones que no pertenecen a las tres categorías anteriores.

La figura 2.2 muestra un ejemplo de conceptos de SNOMED y sus relaciones definitorias.

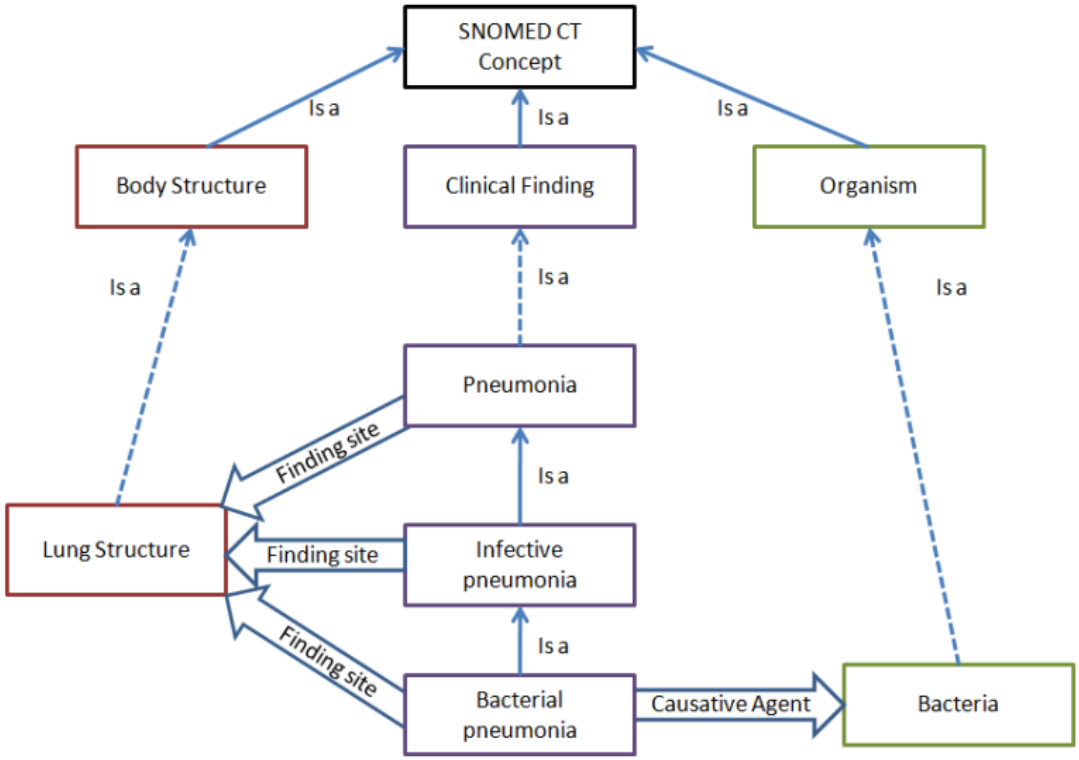

Figura 2.2: Ejemplo de conceptos y sus relaciones definitorias

2.1.1.2 Forma normal SNOMED

La forma normal [3] consiste en una representación común de conceptos biomédicos que puede generarse para cualquier expresión válida aplicando un conjunto de reglas de transformación lógicas.

La principal característica de la forma normal de SNOMED es que todos los conceptos presentes en esta representación son primitivos, es decir, sus padres y atributos no expresan completamente su significado.

Estructura

La forma normal se estructura en dos partes:

- **Focus concept:** El focus concept (concepto foco) se refiere al padre primitivo más cercano del concepto del que se desea encontrar la forma normal.
- **Refinement:** Esta parte la componen las relaciones definitorias del concepto del que se busca la forma normal. Estas relaciones pueden estar agrupadas en cero o más "Role Groups". El mecanismo de Role Groups agrupa las relaciones según la procedencia de las mismas permitiendo una mayor facilidad de lectura y análisis de las mismas.

Según las relaciones que se incluyen en el "Refinement" existen dos tipos de forma normal:

- **Forma normal larga:** Incorpora las relaciones definitorias del concepto del que se calcula la forma normal y las del Focus concept que se obtiene del mismo.
- **Forma normal corta:** Incorpora únicamente las relaciones del concepto del que se calcula la forma normal.

Como podemos observar la diferencia entre las dos formas normales no es muy extensa. Si bien la forma normal larga proporciona más información de forma directamente legible, dicha información está implícita en la forma normal corta ya que esta incorpora el Focus concept del cual se calcula la forma normal.

Para ilustrar de una forma más visual la estructura de la forma normal, se incluye la figura 2.3 en la que se puede observar que la forma normal presenta una estructura recursiva, es decir, las expresiones en la forma normal se pueden anidar en el campo del valor de las relaciones. Esta estructura de la forma normal conlleva que el proceso de obtención de la misma también sea recursivo.

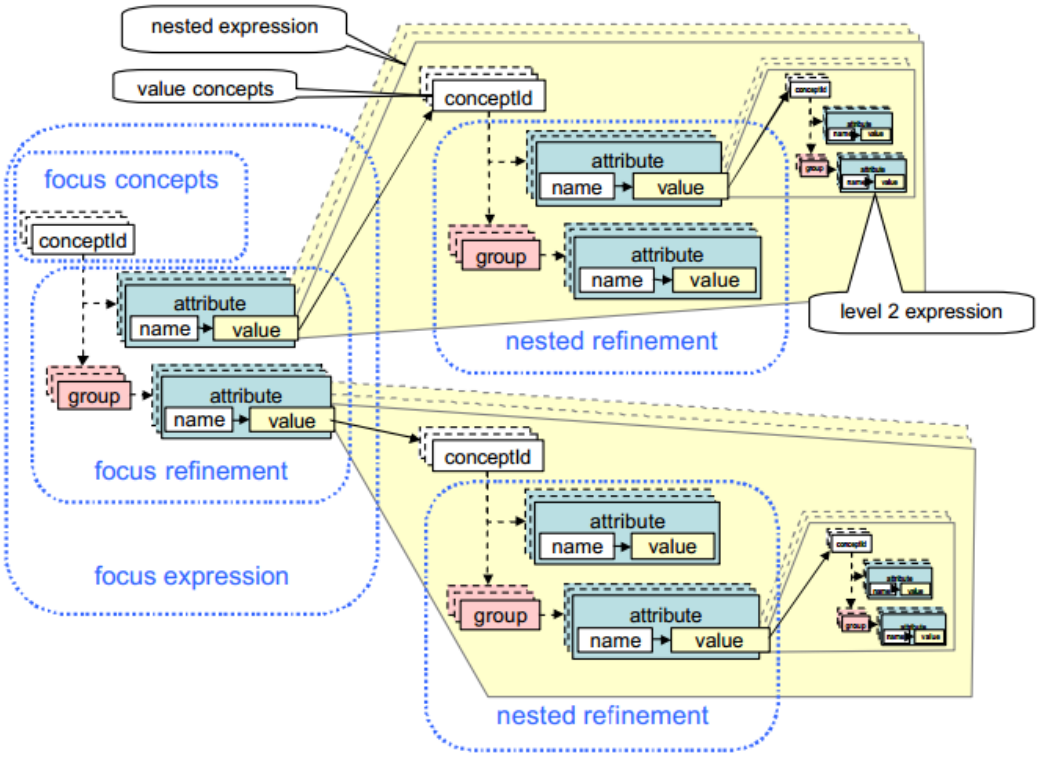

Figura 2.3: Estructura de la forma normal

Proceso

El proceso de obtención de la forma normal se realiza sobre una expresión dada, es decir, un concepto seguido de cero o más relaciones. Este proceso es el siguiente:

1. Obtener el focus concept o primitiva más cercana del concepto de la expresión dada.

2. Obtener las relaciones definitorias del concepto de la expresión dada.

3. El focus concept de la forma normal final será el obtenido en el paso 1, El refinement serán las relaciones obtenidas en el paso 2 más las relaciones que provienen de la expresión de entrada.

4. Se realizará este mismo proceso recursivamente sobre los valores de las relaciones del refinement de la forma normal.

Este proceso nos permitirá obtener la forma normal la cual estará formada exclusivamente por conceptos primitivos.

Ejemplos

A continuación se presentan algunos ejemplos de formas normales que ilustran la estructura y características de la forma normal.

Concept	71620000 | fracture of femur |
Definition *(distributed)*	116680003 | is a | = 64572001 | disease | {116676008 | associated morphology | = 72704001 | fracture | ,363698007 | finding site | = 71341001 | bone structure of femur | }
Long NF *(candidate)*	64572001 | disease | : {116676008 | associated morphology | = 72704001 | fracture | ,363698007 | finding site | = 71341001 | bone structure of femur | }
Short NF *(predicate)*	64572001 | disease | : {116676008 | associated morphology | = 72704001 | fracture | ,363698007 | finding site | = 71341001 | bone structure of femur | }

Figura 2.4: Forma normal de fractura de fémur

En el ejemplo de la figura 2.4 observamos que el focus concept obtenido para la forma normal del concepto Fractura de fémur es Enfermedad o Dolencia (Disease). También observamos que el concepto fractura de fémur tiene dos relaciones, “morfología asociada” y “lugar encontrado” con valores “fractura” y “estructura del hueso del fémur”.

También observamos que las formas normales larga y corta son la misma ya que el concepto “Disease” no incorpora relaciones definitorias propias que se añadan a la forma normal larga.

El ejemplo de la figura 2.5 es similar al anterior ya que tiene el mismo focus concept y las mismas relaciones principales. Sin embargo podemos observar un cambio, ya que el valor de la relación “finding site” es un concepto no primitivo con lo que se debe realizar el proceso de la forma normal sobre ese valor dejando la forma normal del mismo anidado como valor de la relación.

Concept	126716006 | neoplasm of right lower lobe of lung |
Definition *(distributed)*	116680003 | is a | = 126713003 | neoplasm of lung | {116676008 | associated morphology | = 108369006 | neoplasm | ,363698007 | finding site | = 266005 | structure of right lower lobe of lung | }
Long NF *(candidate)*	64572001 | disease | : {116676008 | associated morphology | = 108369006 | neoplasm | ,363698007 | finding site | = (90572001 | structure of lower lobe of lung | : 272741003 | laterality | = 24028007 | right |)}
Short NF *(predicate)*	*Same as long form*

Figura 2.5: Forma normal de concepto con recursividad en su forma normal

La forma normal es la representación seleccionada en los proyectos europeos INTEGRATE y EURECA ya que permite observar los términos simples de los que están compuestos los conceptos médicos complejos permitiendo así realizar cuestiones sobre dichos elementos simples. Por ejemplo se podría preguntar sobre fracturas para que apareciese como resultado fractura de fémur en vez de tener que preguntar por los conceptos en sí.

2.1.2 LOINC

La terminología Logical Observation Identifier Names and Codes (LOINC) [8] incluye una gran base de datos de observaciones para datos médicos de laboratorio. Actualmente incluye más de 50.000 términos accesibles universalmente. Cada uno de estos conceptos incluye seis campos que especifican inequívocamente la prueba, observación o medición. Estos campos son:

- **Componente:** El componente que se mide, evalúa u observa.
- **Tipo de propiedad:** Características de lo que se mide como longitud, masa, volumen,...
- **Aspecto temporal:** Intervalo de tiempo en el que la observación o medición se realizó.
- **Sistema:** contexto o tipo de espécimen con el que se realizó la observación, sangre, orina,...

- **Tipo de escala:** La escala de medición, puede ser cuantitativa, ordinal, nominal o narrativa.
- **Método:** Procedimiento usado para realizar la medición u observación.

Este vocabulario no presenta un mecanismo de composición de términos por lo que no es utilizado en la realización de la herramienta. Sin embargo sigue siendo uno de las terminologías más utilizadas en la actualidad y se incluye en los sistemas de información implementados en los proyectos europeos INTEGRATE y EURECA.

2.1.3 HGNC

La terminología HUGO Gene Nomenclature Committee (HGNC) [9] provee un identificador único y con significado para cada gen humano conocido que en la actualidad alcanza los 35.000 conceptos. También asigna una abreviación a dicho gen para facilitar su lectura. Dichas abreviaturas deben contener únicamente letras latinas y números del 0 al 9.

Esta terminología es básicamente un listado de los genes que incluye relaciones con los grupos de genes en los que se encuentran. Por su naturaleza tampoco existe la composición de conceptos en este vocabulario.

2.1.4 MedDRA

La terminología Medical Dictionary for Regulatory Activities (MedDRA) [10] es utilizada para clasificar información de eventos adversos asociados con el uso de productos farmacéuticos u otros productos médicos. Este vocabulario ha sido desarrollado por el International Conference on Harmonisation (ICH).

Esta terminología se divide en los siguientes elementos estructurales:

- **SOC (System Organ Classes):** Nivel superior de la terminología y se distingue por el sistema anatómico o fisiológico, la etiología o el propósito. Presenta únicamente 26 elementos.
- **HLGT (High Level Group Terms):** Subordinado de SOC y por encima de uno o más HLTs. Presenta 335 elementos.
- **HLT (High Level Group Terms):** Subordinado de HLGT y por encima de uno o más PT. Presenta más de 1.700 elementos.

- **PT (Preferred Terms):** Representa a un solo concepto médico. Contiene más de 19.000 términos.
- **LLT (Lowest Level Terms):** Nivel inferior de la terminología, relacionado con un solo PT como sinónimo o variante léxica. Contiene más de 70.000 conceptos.

La única composición de conceptos que admite esta terminología es la agrupación de términos en un nivel superior por lo que su estudio tampoco está contemplado en la herramienta final de este proyecto.

2.2 Mecanismos de composición

Tras el estudio de las terminologías previamente mencionadas, hemos seleccionado SNOMED como la terminología con las posibilidades más completas para la composición de conceptos complejos. Centrándonos en esta terminología podemos encontrar dos sistemas de composición de conceptos.

2.2.1 Precoordinación

Los conceptos pre-coordinados son aquellos que han sido formados utilizando otros conceptos de la terminología más simples pero que debido a su frecuencia de aparición o utilidad les ha sido asignado un concepto propio el cual incluye implícitamente la información de los conceptos que lo forman. El término se considera "Pre"-coordinado ya que todos los aspectos multifacéticos de un término complejo se pre-coordinan en una forma discreta única.

Este es el tipo de coordinación que hemos observado en las figuras 2.4 y 2.5. Por ejemplo el concepto fractura de fémur es un concepto pre-coordinado ya que tiene asignado un identificador (71620000—Fracture of femur) pero está formado por otros conceptos más sencillos los cuales se han podido observar en la figura 2.4.

Este mecanismo de composición requiere la aceptación por parte de un comité de expertos de la composición de términos y para el estudio de la correcta posición del término en el vocabulario (padres, hijos, relaciones,...). Dada la naturaleza del vocabulario SNOMED, cuyos términos son compuestos siempre por otros más simples (excepto el concepto raíz) todos los términos del vocabulario son pre-coordinados.

2.2.2 Postcoordinación

Por otro lado, el mecanismo de post-coordinación permite componer nuevos conceptos utilizando todos los ya existentes representándolos en un formato que facilita la legibilidad de los mismos. La forma normal nos permite obtener la versión post-coordinada de un concepto pre-coordinado.

Esta forma viene representada con una serie de conceptos simples que se agrupan siguiendo la estructura mostrada en la sección de la forma normal.

Existen tres tipos de post-coordinación:

- **Post-coordinación por refinamiento:** Es un tipo de post-coordinación en el que un concepto se especifica refinando uno de los atributos del concepto. Por ejemplo:

Expression view	Expression
Close-to-user	71620000 | fracture of femur | : 363698007 | finding site | 29627003 | structure of neck of femur |
Normal-form (short or long)	64572001 | disease | : {116676008 | associated morphology | = 72704001 | fracture | ,363698007 | finding site | = 29627003 | structure of neck of femur | }

Figura 2.6: Postcoordinación con refinamiento

  En esta figura hemos podido observar que el valor de la relación "finding site" que por defecto es estructura del hueso del fémur, se refina con una relación adicional en el concepto origen (que es post-coordinado) el cual afecta a su vez a la forma normal cambiando el valor del atributo.

- **Post-coordinación por calificación:** Este es un tipo de post-coordinación en el que un concepto se hace más específico aplicando un valor a uno de los atributos que están permitidos según el modelo del concepto, esto es, que el concepto permita dicha relación calificadora. Por ejemplo:
  En este ejemplo la severidad de la fractura de fémur se establece utilizando la relación calificadora "severity". Esta relación se incluye en la forma normal aunque en una relación separada del "Role Group" principal para notificar que no forma parte de la definición por defecto del concepto.

Expression view	Expression
Close-to-user	71620000 | fracture of femur | : 246112005 | severity | = 24484000 | severe |
Normal-form (short or long)	64572001 | disease | : 246112005 | severity | = 24484000 | severe | {116676008 | associated morphology | = 72704001 | fracture | ,363698007 | finding site | = 71341001 | bone structure of femur | }

Figura 2.7: Postcoordinación por calificación

- **Post-coordinación por combinación:** Este tipo de post-coordinación consiste en combinar varios conceptos en una única forma post-coordinada para expresar un significado conjunto. No se puede considerar que uno de los términos sea un calificador del otro sino que más bien se puede considerar como una conjunción de ambos conceptos.

2.3 Marco del proyecto dentro de los Proyectos Europeos INTEGRATE y EURECA

Los vocabularios comentados en secciones anteriores y los mecanismos de composición de términos son pilares utilizados en la integración semántica de datos clínicos de los proyectos europeos INTEGRATE y EURECA. A continuación se explicará a grandes rasgos las características de dichos proyectos y de la parte de integración de datos de los mismos.

- **INTEGRATE**
  El objetivo del proyecto INTEGRATE [4][5] es el de desarrollar una infraestructura para permitir el intercambio de datos y conocimiento y albergar colaboraciones en investigación biomédica a gran escala.

- **EURECA**
  El objetivo del proyecto EURECA (Enabling information re-Use by linking clinical REsearch and CAre) [6] es el de permitir una conexión segura escalable y consistente de información médica almacenada en sistemas de registro de salud electrónicos con sistemas de información de investigaciones clínicas.

Ambos proyectos son similares dado que ambos intentan permitir un intercambio de datos médicos para facilitar la investigación clínica. Sin embargo hay diferencias en la cantidad de información almacenada y el enfoque de los proyectos. En lo que se

refiere a estos proyectos nos centraremos en la capa de interoperabilidad semántica siendo desarrollada por la UPM y que contiene rasgos comunes en ambos proyectos.

La capa de interoperabilidad semántica de ambos proyectos se encarga de proveer de un acceso homogéneo a los datos almacenados que corresponden a diferentes instituciones médicas. Estos datos en general presentan una alta heterogeneidad, cambiando según los diferentes centros sanitarios. Dada esta dificultad este componente se encargará de tratar los datos en el momento de la inserción en el sistema para que tenga una estructura común y de luego proveer un acceso a los datos según esa estructura.

La estructura de los datos utilizada en los proyectos INTEGRATE y EURECA, denominada Common Data Model (CDM) corresponde con una sección del modelo HL7 Reference Information Model (RIM) [11]. Este modelo divide los datos clínicos en Actos (que pueden ser Observaciones, Procedimientos o Administraciones de substancias) y Entidades (que se corresponde con pacientes, sustancias y otras entidades). Un diagrama más completo del modelo utilizado en estos proyectos se puede observar en la Figura 2.8. En ambos proyectos se utilizan bases de datos MySQL con este modelo para almacenar los datos.

Los conceptos médicos almacenados en estas bases de datos pertenecen a alguno de los vocabularios expuestos anteriormente aunque se contempla que se puedan utilizar términos de nuevos vocabularios en el futuro. En el caso de los conceptos de SNOMED-CT debido a su estructura de composición de términos, se deberá obtener la forma normal la cual será almacenada en el modelo [12][13]. Al almacenarse términos simples, es más sencillo crear una base común sobre la que preguntar datos. Es decir, es más sencillo preguntar por términos sencillos comunes en la terminología médica actual que preguntar por conceptos complejos que pueden tener diferentes interpretaciones.

Mediante la creación de esta herramienta se podrá observar estas composiciones de una forma visual y clara que facilitará la comprensión del trabajo realizado por la capa de interoperabilidad semántica en los proyectos europeos INTEGRATE y EURECA por parte de los usuarios de dicha capa.

2.3.1 Term binding

En los proyectos de investigación previamente mencionados se utilizó un mecanismo para asociar los conceptos médicos de las terminologias con el componente del

modelo HL7 RIM correspondiente. Este proceso es denominado Term binding.

El proceso del Term binding realiza un análisis de la posición del concepto dentro de la jerarquia de términos del vocabulario SNOMED-CT para determinar si se trata de una Observación, un Procedimiento,... Este proceso aún se encuentra en desarrollo pero consigue aproximar de forma efectiva el componente asociado con la mayor parte de conceptos médicos y donde almacenarlos.

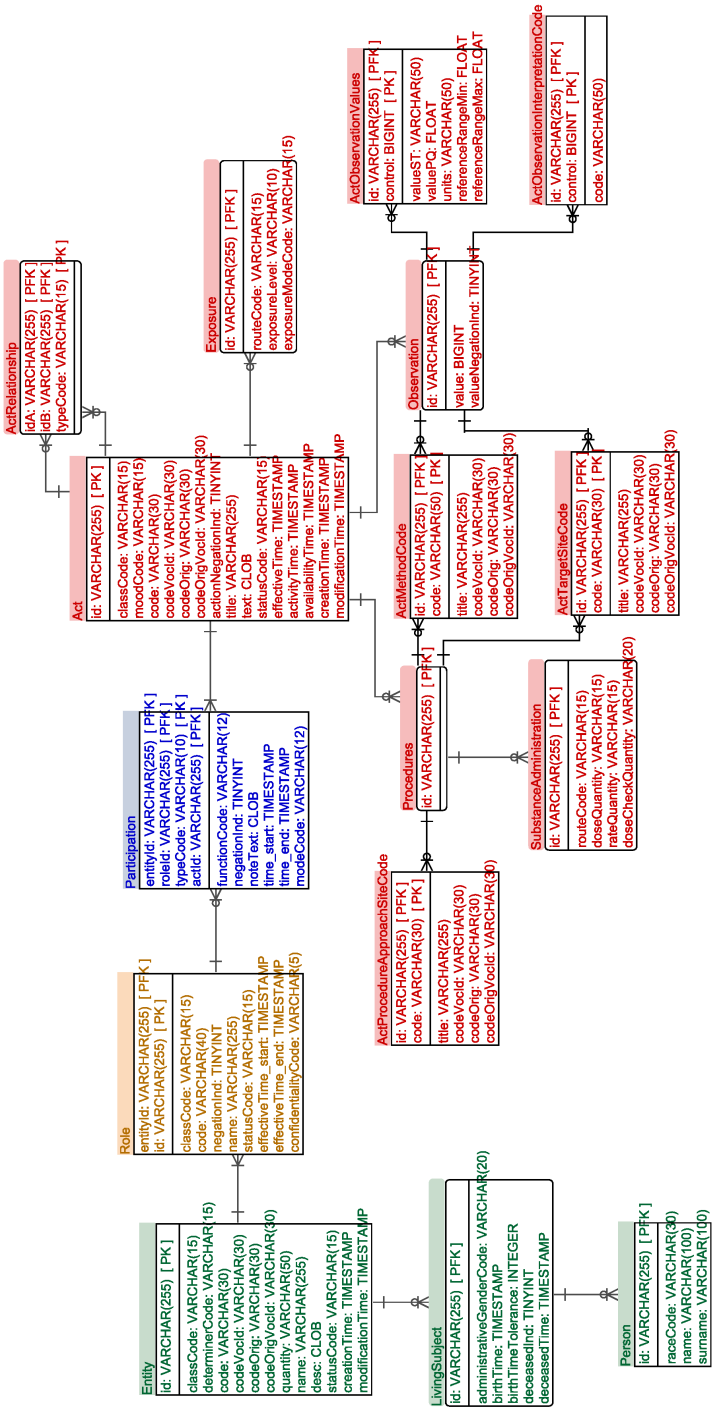

Figura 2.8: Sección HL7 RIM

Capítulo 3

TECNOLOGIAS EMPLEADAS

Durante la realización de la herramienta se han utilizado diversas tecnologías como base para el funcionamiento de la misma. Dada la orientación web de la herramienta muchas de estas tecnologías están enfocadas a ser utilizadas a través de Internet. Se han estudiado alternativas para diversas de estas herramientas y tecnologías las cuales se explicarán en las correspondientes secciones.

3.1 HTML5

HTML5 [14] es una de las últimas tecnologías en la realización de interfaces y aplicaciones web. Es una evolución de HTML4 aunque normalmente se conoce como HTML5 a la conjunción de las tecnologías HTML5+CSS3+Javascript [15] dada la estrecha dependencia que presentan entre ellas. Estas tres tecnologías se utilizan para tres aspectos del desarrollo claramente definidas:

- **HTML:** HTML es un lenguaje de etiquetado con una serie de etiquetas predefinidas y es utilizado para representar el contenido (la información) de una forma estructurada.
- **CSS:** CSS es un lenguaje que permite aportar estilo a la información contenida en el HTML
- **Javascript:** Javascript es un lenguaje de programación de script que otorga funcionalidades y dinamismo al contenido y el estilo de las aplicaciones web.

A continuación detallaremos cada una de estas tecnologías.

3.1.1 HTML

HTML (HyperText Markup Language) [14] permite estructurar el contenido utilizando una serie de etiquetas predefinidas. HTML permitió establecer estándares a seguir a la hora de representar información a través de protocolos HTTP mediante internet. Un ejemplo de la estructura base de HTML se muestra en la figura 3.1:

```
1 <html>
2    <head>
3   Aqui va titulo, dependencias, metadatos...
4    </head>
5    <body>
6   Aqui va el contenido de la pagina
7    </body>
8 </html>
```

Figura 3.1: Estructura básica HTML5

Como podemos observar HTML sigue una estructura prácticamente idéntica a XML y existen versiones de HTML que siguen completamente el estándar XML. Estos documentos son procesados por el navegador cliente para mostrar la web de la forma establecida.

La estructura anterior tiene como etiqueta principal <html> la cual indica que se trata de un documento HTML5. Versiones anteriores de HTML incorporaban más información referente a la versión concreta de HTML utilizada pero en HTML5 se decidió simplificarlo dejando únicamente la etiqueta sin atributos adicionales.

Dentro de esta etiqueta se realiza una división en dos secciones <head> y <body>. La sección "head" suele contener el título del documento HTML, una descripción del contenido del mismo y las dependencias con ficheros externos que pueda contener, es decir, la inclusión de ficheros CSS o Javascript. Estos últimos también se colocan en ocasiones al final del documento HTML. Esto es debido a que al procesar el fichero HTML los ficheros con dependencias se descargan desde el servidor de forma bloqueante con lo que ante una gran cantidad de ficheros Javascript o ficheros grandes, el procesado de la página en sí se ve retrasado hasta completar la descarga de todos los ficheros. Por otro lado, la sección "body" posee toda la información del documento HTML. Es en esta sección donde se introducen las diferentes etiquetas que permiten estructurar el contenido y que luego serán estilizadas mediante CSS. Siguiendo el estándar XML todas estas etiquetas se pueden anidar

permitiendo crear estructuras complejas. A continuación se expondrán algunas de las etiquetas HTML utilizadas para realizar la herramienta:

Etiqueta	Descripción
<table>	Permite creación de tablas
<p>	Su contenido se considera un párrafo del texto
	Permite realizar una división del contenido respecto al resto de forma abstracta pero controlable mediante CSS
<hX>	Permite definir cabeceras de texto. (X es un número de 1 a 6 siendo 1 un título del más alto nivel y 6 del mínimo)
<script>	Incluye código javascript
<img>	Permite incluir imágenes en el documento
<form>	Indica que el contenido de la etiqueta es un formulario

Cuadro 3.1: Etiquetas HTML

Por supuesto estas son solo algunas de las etiquetas utilizadas en la herramienta. Una de las novedades con HTML5 es la inclusión de etiquetas con un componente más semántico con etiquetas para mostrar direcciones, teléfonos, información de las personas,... También se incorporaron APIs para el acceso a herramientas útiles como puede ser el "localStorage" o almacenamiento local que permite utilizar las aplicaciones hasta cierto punto de forma offline. Otro ejemplo sería la API de mapas que permite utilizar el servicio preferido para visualizar localizaciones geográficas. También se han incluido elementos que permiten visualizar videos sin necesidad de herramientas externas como Flash o realizar diagramas de forma dinámica. Esto último es utilizado en esta herramienta a través del framework Raphael.js y Joint.js.

Todas estas novedades han convertido a HTML5 en la norma a seguir para la creación de páginas web aunque el estándar HTML5 aun no ha sido aprobado oficialmente por el W3C (World Wide Web Consortium). Este hecho conlleva que muchos navegadores aún no sean compatibles con muchas de las funcionalidades de HTML5 lo que a su vez provoca diferentes visualizaciones o mal funcionamiento en navegadores sin dichas características. Durante la realización de la herramienta los navegadores de prueba principales han sido Chrome y Firefox.

3.1.2 CSS

CSS (Cascade Style Sheet) [16] es la tecnología predominante a la hora de asignar estilo al contenido de los documentos HTML. En su versión 3, CSS presenta un formato similar al del ejemplo de la figura 3.2.

```
1  .search-query {
2    padding: 4px 14px;
3    margin-bottom: 0;
4    font-family: "Helvetica Neue", Helvetica, Arial, sans-serif;
5    font-size: 13px;
6    font-weight: normal;
7    line-height: 1;
8    -webkit-border-radius: 15px;
9    -moz-border-radius: 15px;
10   border-radius: 15px;
11 }
```

Figura 3.2: Ejemplo CSS

Existen tres métodos para incluir formato CSS en HTML.

- **Inline:** El código CSS se introduce en el propio documento HTML mediante el atributo "style" en la etiqueta HTML del elemento al que se quiere aplicar el estilo. Este método es aplicable para aplicar estilos únicos y muy puntuales.
- **Internal Style Sheet:** Se introduce el código CSS dentro de una etiqueta <style> en el documento HTML. Es correcta pero el código CSS no es reutilizable en otros documentos HTML.
- **External Style Sheet:** El código CSS se encuentra en su propio fichero que puede ser referenciado por múltiples páginas HTML. Es la forma más utilizada y preferida ya que separa completamente el contenido del estilo y permite la reutilización de los estilos. La dependencia desde el código HTML se establece mediante la etiqueta <link>.

En el ejemplo de la figura 3.2 podemos observar la estructura del CSS. Esta estructura presenta un selector (en el ejemplo ".search-query") y entre llaves una serie de declaraciones con formato propiedad-valor ("font-size" la propiedad y "13px" valor). Los selectores permiten seleccionar los elementos HTML a los que queremos aplicar el estilo. Existen dos tipos de selectores.

- **id:** Sirve para seleccionar un elemento HTML de una forma única. En el documento HTML se le añade un atributo "id" al elemento a seleccionar. Este "id" debe ser único en todo el documento HTML. En el CSS, la forma de seleccionar utilizando un id es utilizando el selector "#nombreId".

- **class:** Nos permite seleccionar varios elementos sobre los que aplicar el estilo. Se añade un atributo "class" a los elementos a seleccionar (que tengan mismo estilo). En el CSS, la forma de seleccionar utilizando un class es mediante el selector ".nombreClass"

Existen más formas de seleccionar elementos. Un ejemplo sería "div p" el cual seleccionaría todos los elementos "p" que sean hijos de un elemento "div". También existe "p.nombreId" que seleccionaría todos los elementos "p" cuyo atributo class sea "nombreId".

Existe la posibilidad que dos reglas CSS como las mostradas en el ejemplo afecten a algún elemento en común. En estos casos si las propiedades establecidas son distintas se aplicarían todas y en el caso de solapamiento se aplicaría la más específica.

3.1.3 Javascript

Javascript [17] es un lenguaje de scripting utilizado ampliamente en el entorno del desarrollo web. Este lenguaje se encarga de otorgar dinamismo y capacidades avanzadas a las interfaces creadas mediante HTML y CSS. También es capaz de encargarse del procesamiento de los formularios y de ciertas comunicaciones con el servidor. En el pasado Javascript se instauro como un lenguaje utilizado en la parte cliente de las páginas web (los navegadores) dejando las tareas relacionadas con el servidor a otros lenguajes como PHP. Esta tendencia ha cambiado con la introducción de HTML5 y las ampliaciones de Javascript que permiten que este lenguaje sea utilizado para realizar cualquier función necesaria de la aplicación web. Se trata de un lenguaje interpretado, es decir, no requiere compilación y el procesamiento se realiza cuando es necesario (cuando el código correspondiente se ejecuta). No es un lenguaje fuertemente tipado y permite almacenar cualquier tipo de elemento en cualquier tipo de variable.
Algunas de las características del lenguaje Javascript son:

- **Interpretado:** no requiere compilación y el procesamiento se realiza cuando es necesario (cuando el código correspondiente se ejecuta).

- **No tipado:** No es un lenguaje fuertemente tipado y permite almacenar cualquier tipo de elemento en cualquier tipo de variable.
- **Orientado a objetos:** Incorpora los objetos como entidades que permiten agrupar información y funciones. Esto se vuelve especialmente útil dadas las características de Javascript que permiten almacenar funciones en variables que a su vez permiten una mayor reutilización del código en diferentes clases.

El uso de Javascript suele estar acompañado del uso de la librería JQuery [18]. Esta librería presenta varias funcionalidades que aumentan drásticamente la sencillez y utilidad de Javascript. Algunas de estas funcionalidades son:

- **Selectores CSS:** Permite seleccionar elementos del HTML utilizando los mismos selectores utilizados con CSS y descritos en la sección 3.1.2.
- **Manipulación de elementos DOM:** Un elemento DOM (Document Object Model) corresponde con una etiqueta introducida en el documento HTML. JQuery permite la modificación de dichas etiquetas añadiendo o quitando atributos, añadiendo o quitando o moviendo etiquetas hijas o padres,... Esto presenta un gran potencial ya que permite modificar los valores de los atributos id o class de cada etiqueta cambiando el aspecto asociado por el CSS o modificar lo que se muestra al cliente dinámicamente. También permite gestionar de forma sencilla eventos producidos por los usuarios.
- **Tratamiento de eventos:** JQuery también incorpora funciones que facilitan el tratamiento de los eventos producidos por los usuarios mediante la interacción con la interfaz.

Todas estas funcionalidades serán utilizadas frecuentemente en la herramienta para permitir la actualización dinámica de los contenidos y la interfaz de acuerdo a las búsquedas de los usuarios.

Como hemos comentado Javascript ha sido utilizado típicamente en el lado del cliente en las comunicaciones, sin embargo han surgido opciones para utilizar Javascript también en el lado del servidor. Es el caso de Node.js que permite crear un servidor con comunicación con el cliente de forma rápida utilizando únicamente el lenguaje Javascript. Este sistema esta muy bien integrado con el uso de servicios REST y bases de datos NoSQL como MongoDB [19] y se explicará en mayor detenimiento en la sección 3.3.

3.2 Frameworks

Dada la complejidad de la herramienta y para evitar volver a escribir código ya disponible se han utilizado diversos frameworks (o librerías) que facilitan la elaboración de la aplicación. Estas librerías son:

3.2.1 Bootstrap

Bootstrap [20] es un framework para las interfaces web. Creado por Twitter, permite la creación de interfaces web estilizadas de una forma rápida y sencilla. Básicamente consiste en un archivo CSS y un archivo Javascript que implementan cualidades gráficas a los elementos establecidos en los HTML asociados. De esta forma una tabla normal en HTML puede convertirse en una gráfica estilizada y con efectos dinámicos simplemente introduciendo en el atributo class de la tabla el valor "table".

Esta característica ya facilita suficientemente la creación de un diseño propio para la creación de interfaces web pero otra característica especialmente útil de este framework es la capacidad de crear interfaces "responsive". Esto quiere decir que la interfaz es adaptable a la interfaz utilizada por los usuarios, es decir, adaptable a los diferentes tamaños de pantalla incluyendo móviles y tablets. Crear una interfaz *responsive* es tan sencillo como indicar unas clases especiales en el atributo class de los elementos que organizan la página.

Estas razones junto con el cuidado estético obtenido en el producto final son la razón que se haya decidido utilizar este framework actualmente uno de los más utilizados por los desarrolladores.

3.2.2 Backbone

Backbone [22] es un framework que permite seguir un modelo de arquitectura MVC (Modelo-Vista-Controlador) [21] en la creación de aplicaciones web. Es altamente soportado por servicios REST pudiendo utilizarlos para cargar los modelos de forma sencilla, únicamente indicando las URL involucradas con los recursos disponibles por los servicios REST.

El modelo de arquitectura MVC intenta dividir los componentes que contienen la información (modelo) de los componentes involucrados con la representación gráfica de los mismos (vistas) y la interacción con esos componentes gráficos (controladores).

Estos tres componentes se comunican entre si siguiendo las interacciones de la figura 3.3.

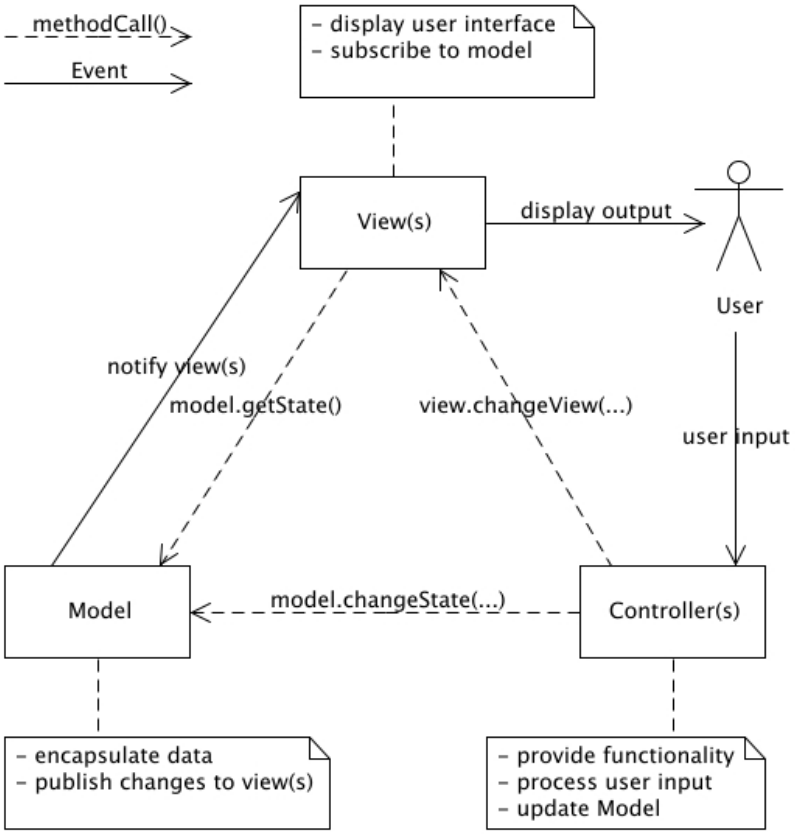

Figura 3.3: Estructura e interacciones de la arquitectura MVC

La arquitectura MVC es una de las más utilizadas dado que la organización que propone es eficaz para poder dividir la complejidad a la hora de crear interfaces gráficas (web o de otro tipo).

Backbone es un framework que permite trasladar esta arquitectura de forma rápida a la web. Incluye definiciones de tipos Javascript para modelos, vistas, colecciones de modelos y otros. Estas definiciones incluyen los campos de datos y las funciones necesarias para la interacción de los módulos y una interacción sencilla con los servicios REST. Por ejemplo asociar un modelo a un recurso proporcionado por una operación GET de un servicio REST es tan fácil como asociar la URL asociada

a el campo "url" del modelo. También se puede asociar una URL con una colección de modelos indicando que devuelve un conjunto de modelos.

3.2.3 Underscore

Underscore [23] es un framework de apoyo que provee una serie de funciones adicionales muy interesantes a Javascript. También es básico para el funcionamiento de Backbone ya que utiliza varias de esas funciones.

Una de las principales utilidades de Underscore es el uso de templates de código HTML. Las templates permiten definir una estructura HTML que deberán seguir los datos procesados mediante Javascript. Se muestra un ejemplo en la figura 3.4.

```
1 <script type="text/template" id="conceptListTemplate">
2   <ul>
3   <% concepts.forEach (function(c) { %>
4     <li><a href="#<%=c.code%>"><%=c.title%></a></li>
5   <% }); %>
6   </ul>
7 </script>
```

Figura 3.4: Ejemplo template Underscore

En el ejemplo de la figura 3.4 observamos la definición de un script que contiene no código Javascript sino HTML. En ese código existe código Javascript embedido mediante etiquetas adicionales. En este caso tenemos una lista <ul> y recorremos la lista de conceptos para añadir una entrada a la lista por cada uno de esos conceptos. Suponiendo que la lista de conceptos contiene tres términos la operación realizada por Underscore con el template quedaría reflejado en la figura 3.5.

```
1 <ul>
2   <li><a href="#71620000">Fracture of femur</a></li>
3   <li><a href="#254837009">Malignant tumor of breast</a></li>
4   <li><a href="#59441001">Structure of lymph node</a></li>
5 </ul>
```

Figura 3.5: Ejemplo template Underscore procesada

Este mecanismos es especialmente útil para crear las vistas en función de los modelos de conceptos almacenados con lo que este mecanismo se suele utilizar a la

hora de configurar las Vistas creadas por Backbone.

Esta librería también incluye una serie de funcionalidades adicionales útiles para el desarrollo de aplicaciones web involucradas principalmente con el tratamiento de listas (incluyendo colecciones de modelos). Estas operaciones incluyen unión, intersección, diferencia, filtrado, mapeo, reducción y muchas otras que lo convierten en herramienta indispensable a la hora de tratar con listas de objetos.

3.2.4 Raphael

Raphael [24] es una librería para el desarrollo dinámico de gráficos o diagramas en HTML que utiliza la especificación SVG (Scalable Vector Graphics) [25]. Estos diagramas se pueden desarrollar dinámicamente en función de datos provistos por los usuarios y permiten cualquier grado de escalabilidad al tratase de gráficos vectoriales. En el desarrollo de la herramienta este framework ha sido utilizado para la creación dinámica de diagramas mostrando datos relacionales de los conceptos médicos.

Raphael funciona de una forma sencilla. Primero se crea un lienzo de un tamaño dado (más tarde redimensionable). Tras ello se crean los objetos a incluir en el diagrama que pueden ser desde círculos, cajas, a figuras complejas para diagramas y gráficos. Tras esto se puede realizar uniones entre los diferentes objetos. Pese a que Raphael incluye esta funcionalidad, se trata de una funcionalidad absoluta determinando puntos origen y destino de las uniones o flechas. Para ayudar en ese aspecto se utilizó otra librería de apoyo llamada Joint.js [26].

Joint se construye sobre Raphael y permite ampliar las posibilidades en lo referente a la unión de objetos creados en un lienzo Raphael. Esto permite unir dos objetos indicando dichos objetos a una función provista en el framework lo cual facilita bastante la gestión y unión de los objetos en cuestión.

3.3 Node.js

Node.js [27] es una plataforma para construir aplicaciones rápidas y escalables a través de la red. Permite construir estas aplicaciones usando Javascript en el lado del servidor lo que permite una comunicación más directa entre los lados cliente y servidor de las aplicaciones al estar escritos en el mismo lenguaje.

El montaje de un servidor utilizando Node.js es sencillo. Un ejemplo de un típico "Hola Mundo" sería mostrado en la figura 3.6. Este código estaría almacenado en un fichero tras lo cual para empezar el servidor sería suficiente con ejecutar el comando 'node fichero' en una terminal o ventana de comandos. En el ejemplo simplemente se obtiene un "Hello World" como resultado.

```
1 var http = require('http');
2 http.createServer(function (req, res) {
3   res.writeHead(200, {'Content-Type': 'text/plain'});
4   res.end('Hello World\n');
5 }).listen(1337, '127.0.0.1');
6 console.log('Server running at http://127.0.0.1:1337/');
```

Figura 3.6: Hola Mundo Node.js

Al tratarse de un servidor en Javascript la interoperabilidad con otros servicios REST o la generación de unos propios se realiza de forma sencilla si bien normalmente se hace uso de Express.js, un framework de Node.js que facilita aún más dicha gestión de servicios permitiendo una orquestación de servicios complejos. Toda la transferencia de datos se realiza en JSON que está bastante integrado con los modelos de Backbone y los diccionarios de Javascript en contraste con XML que presentaría una serie de problemas de parseo.

3.3.1 Servicios REST

Los servicios REST (Representational State Transfer) [28] permiten realizar una serie de operaciones sobre un conjunto de recursos dados. Para ello hace uso de las operaciones HTTP indicadas en la tabla 3.2. Estas operaciones son denominadas operaciones CRUD (Create, Read, Update, Delete)

Operación	Descripción
GET	Obtención de un recurso
POST	Creación de un nuevo recurso
PUT	Actualización de un recurso existente
DELETE	Borrado de un recurso

Cuadro 3.2: Operaciones REST

Estas operaciones se realizan sobre los recursos, cada uno de los cuales debe poseer una URI (Uniform Resource Identifier) única sobre la que se realizarán las operaciones.

Las operaciones anteriores poseerán una serie de resultados esperados (200 OK, 404 Not found,...). Las URIs de los recursos, las operaciones disponibles sobre ellos y los posibles resultados obtenidos se deben definir para permitir a un desarrollador hacer uso completo de dichos servicios REST. La definición de los servicios REST creados para la herramienta está especificada en la sección 4.3.7.

Para generar las funcionalidades de esta herramienta, además de generar servicios REST también consumimos los mismos de forma local. Dada la naturaleza de no manipulación de datos de la herramienta, las operaciones POST, PUT y DELETE que modifican los datos no son necesarias y todos los recursos necesarios se obtendrán a través de operaciones GET.

Una forma rápida de hacer uso de las operaciones GET de los servicios es a través de cualquier navegador, introduciendo la URI correspondiente lo que realiza esta operación que devuelve los datos resultantes.

3.3.2 JSON

JSON (JavaScript Object Notation) [29] es el formato de transferencia de datos utilizado en los servicios REST y se ajusta completamente a la estructura de los diccionarios en Javascript incluyendo los modelos de datos de Backbone. Un objeto Javascript es fácilmente convertible a formato JSON y lo opuesto es igualmente válido. JSON presenta una estructura clave-valor donde el valor puede ser otro JSON. Se muestra un ejemplo de código Javascript en la figura 3.7 .

Como podemos observar, para cada clave un valor es asociado, siendo este otro JSON, un valor (numérico o textual) o un conjunto (o array) de los dos anteriores. Esta estructura recuerda a los objetos en Javascript donde el par clave-valor podría transformarse a un par variable-valor donde cada JSON se consideraría un objeto. Esta versatilidad permite que el tratamiento de los datos a través de JSON se realice de una forma nativa en Javascript evitándonos necesidades de parseo o tratamiento de la información transferida.

```
{"menu": {
   "id": "file",
   "value": "File",
   "popup": {
     "menuitem": [
       {"value": "New", "onclick": "CreateNewDoc()"},
       {"value": "Open", "onclick": "OpenDoc()"},
       {"value": "Close", "onclick": "CloseDoc()"}
     ]
   }
}
```

Figura 3.7: Ejemplo JSON

3.3.3 Express

Express.js [30] es un framework para Node.js que añade una serie de interesantes funciones al servidor creado con esta tecnología y que facilita muchas operaciones en el desarrollo del mismo.

Express esta orientado al desarrollo de servicios web y permite crear servicios REST con mucha facilidad utilizando Javascript como se muestra en la figura 3.8.

```
var express = require('express');
var app = express();
app.get('/:name', function(req, res){
  res.send('Hello '+req.params.name);
});
app.listen(3000);
console.log('Listening on port 3000');
```

Figura 3.8: Ejemplo servicio creado con Express

En la figura 3.8 se muestra la creación de un servidor con Node.js en el puerto 3000 y la implementación de un servicio en dicho servidor que devuelve “Hola tunombre” tras hacer una consulta GET con “tunombre” (ej. http://localhost:3000/Pedro devolvería “Hello Pedro”).

3.3.4 Otros framework

Además de Express.js, utilizamos una serie de frameworks adicionales en el servidor en Node.js que facilitan la realización de la herramienta y la obtención de datos. Estos framewok son:

- **Framework SPARQL:** Este framework permite realizar consultas de datos utilizando el lenguaje de consultas para la información en RDF SPARQL (sección 3.6.3). Esto evita tener que utilizar otro tipo de mecanismos o servicios para el acceso a esos datos. El framework utiliza la interfaz REST del servicio Sesame (sección 3.6) para el acceso a los datos.
- **Framework Require:** Hay cierta información que no se puede acceder directamente sobre los datos provistos por Sesame sino que requieren cierto procesamiento. Este procesamiento, por simplificación, se realiza mediante otros mecanismos. En este caso, la obtención de la forma normal se realiza mediante un Servlet que obtiene la forma normal procesando los datos almacenados en el servicio Sesame. Este servlet puede ser utilizado mediante una interfaz REST pero para hacer uso de la misma es necesario de un framework en Node.js. El framework Require realiza esta función permitiendo utilizar otros servicios REST simplemente indicando URL y la correspondiente operación CRUD.

3.4 Servlet

Un servlet [31] es una clase Java utilizada para extender las funcionalidades de un servidor. Puede responder a diferentes tipos de peticiones. En el caso de esta herramienta se tratará de peticiones REST que conformaran un acceso interno a la información de los vocabularios médicos almacenados en un servicio Sesame (también con acceso REST).

Los servlets permiten crear servicios utilizando el lenguaje de programación Java y son una de las tecnologías más comunes en la creación de estos servicios en la actualidad dada la popularidad del lenguaje de programación Java.

Los servlets son clases Java cuyos métodos se pueden realizar de forma remota. Deben añadirse a un contenedor de Servlets como Tomcat para ser accesibles. Un ejemplo de Servlet se puede observar en la figura 3.9.

En el ejemplo de la figura 3.9 creamos un servlet a través de la clase “restFrontend” utilizando como ruta “/snomed”. Dentro de este servlet creamos un método

```
1 @Path( "/snomed" )
2 public class restFrontend {
3   @GET
4   @Produces( MediaType.APPLICATION_JSON )
5   @Path( "{conceptID}/parents")
6   public static String getParents( @PathParam( "conceptID" ) String
        conceptID){
7     return NormalFormGenerator.getParents(conceptID);
8   }
9 }
```

Figura 3.9: Ejemplo Servlet

que se corresponderá con uno de los recursos disponibles por el servicio REST, en este caso devuelve los padres de un concepto de la terminología SNOMED-CT a través de la ruta "/snomed/conceptID/parents" en formato JSON. Cada método del Servlet se encarga de mandar una respuesta acorde a lo que se especifica. De esta forma el valor devuelto por el método del ejemplo deberá ser un String que siga el estándar JSON.

3.4.1 Java

Java [32] es uno de los lenguajes de programación más utilizados para el desarrollo de aplicaciones y por tanto presenta gran cantidad de soporte y frameworks para su uso. Se trata de un lenguaje orientado a objetos, compilado y se ejecuta sobre una máquina virtual que actúa de intermediaria entre el programa ejecutado y el hardware que lo ejecuta. Esta máquina virtual provocó problemas de rendimiento en las primeras versiones del lenguaje pero ha ido mejorándose continuamente consiguiendo un rendimiento prácticamente nativo. Una de las principales ventajas de esta máquina intermedia es la interoperabilidad entre los diferentes dispositivos en los que se puede conseguir un producto operable con un único código fuente. Java es utilizado en gran cantidad de productos incluyendo el contenedor de Servlets Tomcat.

3.5 Tomcat

Tomcat es un contenedor de Servlets open source creado por la fundación Apache. Un contenedor de servlets es básicamente un servidor que permite cargar diferentes Servlets o servicios de una forma rápida y dinámica.

Tomcat es uno de los contenedores de Servlets más populares y por ello cuenta con gran cantidad de soporte si bien existen alternativas como Jetty que permite una gran portabilidad y escalabilidad a la hora del montaje de Servlets con una configuración inicial mínima. La decisión de utilizar Tomcat se basó principalmente en su soporte más extendido y el hecho de que algunos de los servicios montados requerían procesamiento rápido de datos.

En este proyecto, Tomcat se ha utilizado para el montaje del Servlet utilizado por el Frontend o página web para el acceso a los datos del vocabulario SNOMED-CT y un servicio propio que permite consultas sobre ciertos datos de esta misma terminología, Sesame. Dada la extensión del vocabulario SNOMED-CT el proceso creado por el servicio Sesame para realizar consultas rápidas al vocabulario se convierte en un proceso pesado en el sistema llegando fácilmente a ocupar 3GB de memoria en un computador de sobremesa normal.

3.6 Sesame

Sesame [34] es un framework para la consulta y análisis de información en formato RDF. Contiene un repositorio para almacenar esa informacion en un formato de tripletas ("triplestore"). Este repositorio permite el procesamiento de datos RDF (Resource Description Framework), es decir, permite realizar consultas y modificaciones a datos provistos en formato RDF. Los datos concretos de la terminología SNOMED-CT se pueden obtener en formato OWL (Web Ontology Language) a través de un script Perl desarrollado por el IHTSDO (International Health Terminology Standards Development Organization).

Existen dos accesos a la información contenida en un servicio Sesame:

- **Cliente OpenRDF Workbench:** OpenRDF Workbench es una herramienta que se adjunta junto con el servicio Sesame que permite realizar consultas mediante SPARQL a los datos de tripletas contenidos. Esta herramienta hace uso de los servicios REST provistos por el servicio.
- **Acceso directo a los servicios REST:** Las consultas se realizan directamente sobre uno de los repositorios de información del servicio Sesame mediante los servicios REST que posee. El lenguaje de consulta es SPARQL.

Como hemos comentado, las consultas a los datos contenidos en un repositorio Sesame se realizan en el lenguaje SPARQL que permite realizar consultas sobre los datos en forma de tripletas (sección 3.6.2) lo cual se ajusta al formato en el que están almacenados dado el formato RDF en el que se encuentran.

3.6.1 OWL

OWL (Web Ontology Language) [35] es un lenguaje de ontologías con significado formalmente definido y desarrollado por el W3C (World Wide Web Consortium). Estas ontologías cuentan con diversos mecanismos para la representación de la información y son almacenados como documentos web semánticos. También son compatibles con información en formato RDF.

OWL es utilizado como contenedor de la terminología SNOMED-CT. Este fichero es generado mediante un script Perl provisto por los gestores del vocabulario, el IHTSDO. Unas modificaciones a este script facilitan la lectura de la información en el repositorio semántico Sesame.

- **Información de parentesco:** La información de parentesco de los conceptos (conceptos padres o hijos) es complicada de acceder de una forma rápida dada la posición dentro de la estructura del lenguaje que ocupa. Para mantener la estructura pero facilitar el acceso a esta información se decidió duplicar estos datos en una posición más accesible. Esta modificación se realizó para adaptarse a las necesidades de la herramienta por lo que puede no ser necesario en otros casos.

- **Uso de datos inferenciados:** Para evitar un procesamiento adicional en el repositorio semántico creado para almacenar el vocabulario, utilizaremos los datos de la terminología inferenciados. Ambas versiones están presentes en las versiones provistas por el IHTSDO.

- **Inclusión de información adicional:** También aprovecharemos para incluir información adicional que sea útil para el desarrollo de la herramienta como la información de vinculación de los conceptos a el modelo HL7 RIM provista por el proceso del Termbinding (ver sección 2.3.1).

Estos cambios ayudan a que las consultas realizadas al repositorio por la herramienta para determinar la estructura e información de los conceptos se realicen sin complicaciones que deban ser resueltas por un postprocesamiento de esta información.

3.6.2 RDF

RDF (Resource Description Framework) [36] es un framework para la representación de metadatos en la web, creado por el W3C (World Wide Web Consortium). El formato que utiliza para esta representación es XML sobre el que se puede definir un esquema RDFS (RDF Schema). RDF provee un método general y flexible para descomponer cualquier conocimiento en piezas pequeñas llamadas tripletas (triples) con algunas reglas sobre la semántica de esas piezas. Estas tripletas unidas se denominan "Grafo" e incluyen toda la información incluyendo las relaciones entre los elementos individuales de la información

Un ejemplo de uno de los conceptos de SNOMED-CT representado usando los sistemas de representación OWL y RDF viene acompañado en la figura 3.10.

Como podemos observar en la figura 3.10, cada concepto está representado como una clase OWL cuyo identificador sigue la estructura "SCT_CODIGONUM" donde SCT es un identificador para los conceptos de la terminología SNOMED-CT y el CODIGONUM es el código numérico del concepto. Cada concepto contiene una "label" o título, uno o más elementos del terminfo y uno o más padres mediante la etiqueta "rdfs:subClassOf" (excepto el concepto "root"). Los conceptos también incluyen una conjunción (owl:intersectionOf) de restricciones y clases. Estas restricciones corresponden con las relaciones definitorias del concepto y las clases representan los padres del concepto. Esta última información está duplicada para facilitar el acceso mediante queries SPARQL.

Tras haber revisado la estructura del XML podemos definir la estructura interna que tendrán estos componentes al ser procesados. Como hemos comentado RDF descompone el conocimiento en tripletas de información, esto es, Origen-relación-valor. Algunos ejemplos de tripletas provenientes del ejemplo de la figura 3.10 son:

- **SCT_71620000—rdfs:label—Fracture of femur:**
  Esta relación vincula el label (o texto) del concepto con código 71620000 a "Fracture of femur" con una terminología inglesa. Esta relación aparece debido a la línea en xml: <rdfs:label xml:lang="en">Fracture of femur</rdfs:label>

- **SCT_71620000—rdfs:subClassOf—SCT_7523003:**
  Esta relación vincula el concepto con código 71620000 a el concepto con código 7523003 con una relación de parentesco, es decir, 7523003 es padre de 71620000. Esta relación aparece en: <rdfs:subClassOf rdf:resource="SCT_7523003"/>

```
<owl:Class rdf:about="SCT_71620000">
   <rdfs:label xml:lang="en">Fracture of femur</rdfs:label>
   <prv:containedBy rdf:resource="Observation_code"/>
   <rdfs:subClassOf rdf:resource="SCT_7523003"/>
   <rdfs:subClassOf rdf:resource="SCT_46866001"/>
   <owl:equivalentClass><owl:Class>
   <owl:intersectionOf rdf:parseType="Collection">
     <owl:Class rdf:about="SCT_7523003"/>
     <owl:Class rdf:about="SCT_46866001"/>
     <owl:Restriction>
       <owl:onProperty rdf:resource="RoleGroup"/>
         <owl:someValuesFrom>
           <owl:Class>
              <owl:intersectionOf rdf:parseType="Collection">
                <owl:Restriction>
                  <owl:onProperty rdf:resource="SCT_116676008"/>
                  <owl:someValuesFrom rdf:resource="SCT_72704001"/>
                </owl:Restriction>
                <owl:Restriction>
                  <owl:onProperty rdf:resource="SCT_363698007"/>
                  <owl:someValuesFrom rdf:resource="SCT_71341001"/>
                </owl:Restriction>
              </owl:intersectionOf>
            </owl:Class>
         </owl:someValuesFrom>
       </owl:Restriction>
   </owl:intersectionOf>
   </owl:Class></owl:equivalentClass>
</owl:Class>
```

Figura 3.10: Representación del concepto Fractura de fémur en XML

El documento completo de la terminología SNOMED-CT en XML utilizando RDF y OWL ocupa alrededor de 300MB. Dentro del proyecto este archivo se ha ampliado con datos de otras terminologías como LOINC o HGNC si bien la estructura de estos vocabularios es distinta a la que presentan los conceptos de SNOMED-CT. Este archivo es poteriormente procesado por Sesame (sección 3.6) para generar todas las tripletas y contenerlas con un acceso rápido mediante consultas SPARQL.

3.6.3 SPARQL

SPARQL (SPARQL Protocol and RDF Query Language) [38] es un lenguaje creado para la consulta de grafos RDF. Permite el acceso a la información contenida mediante tripletas al proveer un lenguaje que permite la consulta en ese formato. Un

ejemplo de consulta SPARQL se encuentra en la figura 3.11.

```
1  PREFIX prv:<http://purl.org/net/provenance/ns#>
2  PREFIX :<http://www.ihtsdo.org/>
3  PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>
4  PREFIX hl7:<http://test.org#>
5  PREFIX loinc:<http://www.loinc.org/>
6  PREFIX owl:<http://www.w3.org/2002/07/owl#>
7  PREFIX xsd:<http://www.w3.org/2001/XMLSchema#>
8  PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
9  SELECT ?parentsCode ?parentsLabel WHERE {
10     :SCT_71620000 rdfs:subClassOf ?parentsCode.
11     ?parentsCode rdfs:label ?parentsLabel.
12 }
```

Figura 3.11: Ejemplo query SPARQL

Al observar la query SPARQL de la figura 3.11 observamos una serie de prefijos. Estos prefijos determinan ciertos ámbitos de la información y actúan de la misma forma que los prefijos XML. Tras ellos se encuentra la query que sigue el formato: SELECT ?variable... WHERE CONDICIONES_GRAFO . Estas condiciones del grafo siguen la estructura de tripletas. En el caso del ejemplo podemos realizar un análisis de qué tipo de consulta se realiza:

1. Primero obtenemos los padres del concepto con código 71620000 y los referenciamos mediante "?parentsCode". Este paso viene determinado por la línea 10 de la consulta.

2. Obtenemos todas las relaciones que permiten obtener el label o título de los conceptos. Este paso se realiza en la línea 11 de la consulta.

3. Finalmente, dado que el nombre de la variable utilizada como origen en el segundo paso coincide con la variable utilizada para referenciar a los padres del concepto en el paso 1, se restringen los valores posibles en el valor de la relación del segundo paso, obteniendo únicamente los títulos de los padres del concepto 71620000.

4. Seleccionamos el código de SNOMED-CT de los conceptos padres y los títulos correspondientes.

Como hemos observado, con una simple consulta se consigue acceso a los datos obtenidos a través de las relaciones de las tripletas contenidas en el grafo RDF. Este será el mecanismo de consulta utilizado para acceder a toda la información de los vocabularios.

Capítulo 4

DISEÑO E IMPLEMENTACIÓN DE LA HERRAMIENTA

4.1 Especificación de requisitos

4.1.1 Introducción

En esta sección analizaremos los requisitos software determinados para la herramienta así como una serie de análisis del producto que ayudan a determinar dichos requisitos.

4.1.1.1 Propósito

La especificación de requisitos se realiza como una herramienta para una mejor comprensión del producto final por todas las partes involucradas en la elaboración de la herramienta.

Esto incluye una explicación detallada de las funcionalidades esperadas por el sistema y las restricciones necesarias para su correcto uso y funcionamiento. Esta explicación se realizará en un formato comprensible tanto para los desarrolladores como para los usuarios finales de forma que ambas partes tengan un punto en común sobre el que realizar el desarrollo del producto.

4.1.1.2 Ámbito del sistema

El sistema está pensado para que el uso de la misma sea accesible por la mayor parte posible de usuarios, tanto profesionales de la medicina como por usuarios sin conocimientos de la misma.

Pese a que su uso no requerirá conocimientos médicos exhaustivos, una mayor comprensión de los datos mostrados se traducirá en unas mejores conclusiones de los datos obtenidos, es decir, un usuario especializado podrá comprender los resultados y realizar conclusiones de forma más efectiva que un usuario sin conocimientos del vocabulario o su estructura.

El acceso a esta herramienta tampoco reportará problemas si se cuenta con una conexión a internet ya que la herramienta está disponible mediante una web accesible por dispositivos móviles como tablets y smartphones. Esto amplia exponencialmente el número de posibles usuarios y de situaciones en las que es posible el uso de la herramienta.

En definitiva, esta herramienta permitirá realizar un análisis de términos compuestos en un vocabulario médico de gran extensión como es SNOMED-CT de una forma sencilla y será especialmente útil con usuarios que trabajen con dicha terminología como profesionales de la medicina o desarrolladores que trabajen con dicho vocabulario para el desarrollo de otro productos.

4.1.1.3 Definiciones, Acrónimos y Abreviaturas

- **Definiciones:**

Concepto	Definición
SNOMED-CT	Terminología médica con mecanismo de términos compuestos.
Node.js	Plataforma para construir aplicaciones web utilizando Javascript en el servidor
Frontend	Parte del servicio ejecutada en el cliente y generalmente visible e interactuable por el usuario del producto
Backend	Parte del servicio que realiza las operaciones en comunicación con el Frontend.

- **Acronimos:**

Acrónimo	Forma extendida
SNOMED-CT	Systematized Nomenclature of Medicine – Clinical Terms
ERS	Especificación de requisitos software
HTTP	HyperText Transfer Protocol
API	Application Programming Interface
REST	Representational State Transfer
HTML	HyperText Markup Language
URL	Uniform Resource Locator

4.1.1.4 Referencias

IEEE Recomendad Practices for Software Requierements especification ANSI/IEEE 830 1998.

4.1.1.5 Visión general

La especificación de requisitos consta de tres secciones:

- **Introducción:** Aquí introduciremos la ERS incluyendo sus objetivos y una descripción general de la misma.
- **Descripción general:** En esta sección se describe de forma general la herramienta. Introduciremos el producto final explicando las funciones que debe cumplir, restricciones y los datos asociados a cada función.
- **Requisitos específicos:** Una especificación detallada de los requisitos software que debe cumplir la herramienta.

4.1.2 Descripción General

En esta sección describiremos aquellos factores que afectan a la herramienta y a sus requisitos, sin entrar a describir estos últimos. Estos factores determinan el contexto de desarrollo del producto y ayudan a entender los posteriores requisitos.

4.1.2.1 Perspectiva del producto

El producto deberá poder usarse de forma independiente a cualquier otro sistema siempre que se cuente con un dispositivo con un navegador HTTP con acceso a Internet y soporte a Javascript. La API REST por otro lado solo requerirá acceso a Internet para ser usado.

4.1.2.2 Funciones del producto

Las funciones que debe llevar a cabo el producto se pueden resumir en las siguientes:

- **Consulta de datos de conceptos médicos:** Los conceptos médicos en terminologías compuestas como SNOMED-CT presentan cierta información extrapolable a través de sus relaciones con otros conceptos. La herramienta deberá proporcionar un mecanismo útil y rápido para el acceso y análisis a esa información.
- **Análisis de las relaciones entre conceptos:** La herramienta deberá realizar un análisis de las relaciones de los conceptos médicos permitiendo extrapolar dicha información a una forma normal que facilite la comprensión de los términos compuestos.
- **Acceso universal:** La herramienta deberá ser accesible a través del mayor rango posible de dispositivos y para el mayor número de usuarios. Para ello se utilizarán las últimas tecnologías web que permiten la máxima compatibilidad tanto con dispositivos de sobremesa como con dispositivos portátiles.

4.1.2.3 Características de los usuarios

La herramienta está pensada para que su uso sea lo más sencillo posible y que alcance a el mayor número posible de dispositivos. Esto permite que haya una amplia diversidad en los posibles usuarios de la misma. A pesar de esto se pueden diferenciar dos tipos principales de usuarios que realizarán un uso normal del producto:

- **Personal sanitario o de investigación médica:** El personal especializado necesita, de forma frecuente, una terminología común sobre la que representar conceptos médicos. Dado que esta profesión no suele manejar estructuras de datos como la presente en el vocabulario SNOMED-CT, esta herramienta les permitirá realizar una identificación de términos médicos y un análisis de los mismos de una forma sencilla y rápida.
- **Desarrolladores de productos médicos:** Los desarrolladores de productos software relacionados con la medicina suelen necesitar una terminología

esquematizada para representar los conceptos médicos. Bien para realizar consultas relacionadas con los conceptos de forma rápida y visual o para obtener información de los conceptos de forma programática a través de la API REST, esta herramienta debe poder cumplir dichas necesidades.

4.1.2.4 Restricciones

El producto está desarrollado con un enfoque web y utiliza los estándares HTML5 con lo que desde el punto de vista de los usuarios no existen restricciones más que con la de contar con un navegador web con acceso a internet.

Desde el punto de vista de la plataforma del producto no existen restricciones en cuanto a sistema operativo. Esto se debe a que el servidor principal es un servidor Node.js con compatibilidad con todas las plataformas actuales más conocidas. Como hemos comentado en la sección 3.3, este servidor es básicamente un fichero (o varios) de código javascript con lo que es fácilmente portable. También tenemos los servicios de Sesame y los Servlets creados para el tratamiento de la información los cuales deben contenerse en un contenedor de Servlets como Tomcat el cual es multiplataforma.

Dada la naturaleza pública de la información de los vocabularios médicos no existe necesidad de realizar un registro de los usuarios para su acceso a la información con lo que no existen restricciones en ese ámbito. Por otro lado, se debe evitar la modificación de la información de los vocabularios contenida en el servicio Sesame dado que las únicas actualizaciones permitidas se realizarán según las últimas versiones de los respectivos vocabularios y no por los usuarios de la herramienta. Por ello el acceso al servicio Sesame no se puede realizar directamente sino que debe hacer a través de los servicios establecidos en el servidor de Node.js.

4.1.2.5 Suposiciones y Dependencias

- **Suposiciones**
  Suponemos que los requisitos descritos en las siguientes secciones conservarán su validez a través de posibles actualizaciones futuras. También suponemos que los datos de los vocabularios permanecen estables excepto por las nuevas versiones presentadas periódicamente. Tras una nueva actualización de los vocabularios, en concreto SNOMED-CT, será necesaria la obtención de un nuevo OWL que siga la estructura establecida e incluya los nuevos datos.

- **Dependencias**
  Una de las principales dependencias presentes en este producto se encuentra en el uso de una gran cantidad de APIs externas. Estas APIs facilitan enormemente el desarrollo del producto y permiten realizar una serie de tareas complejas de forma sencilla si bien los cambios en dichas APIs pueden alterar el funcionamiento del producto e incluso ser necesaria una actualización de la misma. Siempre existe la posibilidad de no actualizar estas APIs pero las actualizaciones presentadas pueden contener soluciones a problemas de seguridad y rendimiento que es mejor incluir además de que el soporte a versiones antiguas de los productos suele disminuir e incluso desaparecer con el tiempo. De esta forma algunas de las APIs cuyos cambios pueden afectar al funcionamiento de la herramienta serían los frameworks de Express.js, Backbone.js, JQuery, etcétera.

  Además de estas dependencias con las APIs utilizadas, pueden existir dependencias con los estándares utilizados a largo plazo. Si bien se ha intentado utilizar las últimas tecnologías y estándares presentes en el momento del desarrollo de la aplicación, es de suponer que nuevas tecnologías o versiones de las tecnologías actuales se presenten de forma continua lo que conllevará una necesidad de actualización del producto para adaptarse a dichos cambios. Algunas de estas tecnologías son HTML5 (aún no estandarizado), Node.js (sin estándares), OWL, RDF, ectétera

4.1.3 Requisitos Específicos

En esta sección se presentarán de forma concreta los requisitos que afectan a la interfaz del usuario y el funcionamiento de la herramienta incluyendo los estándares a seguir por la misma.

4.1.3.1 Interfaces externas

Este tipo de requisitos definirán el tipo de interacción y las funcionalidades disponibles en las interfaces del producto con los usuarios.

Interfaz hardware
No definida

Interfaz software

Como hemos comentado, la interfaz será realizada utilizando tecnologías web para permitir el acceso al mayor número posible de dispositivos e usuarios. Los requisitos establecidos a cumplir por la interfaz del usuario son:

- **REQ#01:** Se dispondrá de una interfaz que permita realizar búsquedas rápidas de conceptos de las terminologías incluidas.
- **REQ#02:** El sistema permitirá cambiar la visualización entre los datos de los diferentes conceptos buscados de una forma visual e intuitiva.
- **REQ#03:** La navegación descrita en el requisito REQ#05 no debe impedir visualizar el concepto original.
- **REQ#04:** El sistema debe permitir dejar de visualizar los conceptos buscados una vez dejen de ser útiles.

4.1.3.2 Funciones

Las funciones de la herramienta están relacionadas con la información mostrada de los vocabularios médicos y las funcionalidades que deberá permitir según esa información.

- **REQ#05:** El sistema deberá permitir navegar entre los conceptos presentes en las relaciones de los conceptos buscados incluyendo padres, hijos, relaciones definitorias y relaciones en los que aparecen de una forma visual.
- **REQ#06:** Las búsquedas descritas en el requisito REQ#01 se deberán poder realizar utilizando el nombre completo del concepto o el código asociado en la terminología.
- **REQ#07:** Cuando la búsqueda se realice según el nombre completo del concepto, el sistema deberá proporcionar sugerencias sobre el nombre del concepto buscado.
- **REQ#08:** Tras la selección del concepto, el sistema deberá obtener los datos de las relaciones de dicho concepto y la forma normal de SNOMED-CT (en su caso). Esto incluye padres e hijos del concepto, relaciones definitorias del mismo y su aparición en relaciones definitorias de otros conceptos.
- **REQ#09:** El sistema permitirá la búsqueda y análisis de múltiples conceptos.

4.1.3.3 Requisitos de rendimiento

Dada la naturaleza web del producto, las conexiones a la red influyen en gran medida con su rendimiento. También se debe considerar la gran cantidad de conceptos almacenados a la hora de realizar las búsquedas. Por tanto se establecen los siguientes requisitos que afectan al rendimiento de las funciones descritas en secciones anteriores:

- **REQ#10:** La búsqueda y análisis de un concepto deberá realizarse en un tiempo mínimo que permita realizar dichas operaciones y un máximo que no produzca impaciencia en los usuarios.
- **REQ#11:** El sistema deberá permitir el uso por parte de múltiples usuarios sin una penalización notable en su rendimiento.
- **REQ#12:** El sistema de acceso a la información contenida deberá realizarse lo más rápidamente posible.

4.1.3.4 Restricciones de Diseño

Estos requisitos especifican restricciones de diseño impuestas por estándares, plataformas hardware y software, etc. Como hemos comentado previamente, el servidor Node.js y el contenedor de Servlets son tecnologías altamente portátiles lo que evita restricciones en lo referente al sistema operativo.

Por otro lado la inferencia realizada por el servicio Sesame en los datos de las terminologías realiza una carga de todos dichos datos en la memoria del sistema. Esto provoca que la máquina a alojar el servicio deba contener una cantidad razonablemente grande de memoria.

- **REQ#13:** La plataforma de acceso a la herramienta será a través de cualquier navegador a través de Internet.
- **REQ#14:** Se utiliza Node.js como plataforma de montaje de la herramienta y Javascript como el lenguaje de programación asociado.
- **REQ#15:** Los datos manejados en las interacciones entre el cliente y el servidor y el propio almacenamiento y procesamiento en ambas partes se realizarán en formato JSON.
- **REQ#16:** El sistema obtendrá los datos a través de una API REST.

4.1.3.5 Atributos del Sistema

En este apartado se detallan los atributos de calidad del sistema: seguridad, fiabilidad, mantenibilidad y portabilidad. Se especifican los tipos de usuarios que están autorizados, o no, a realizar ciertas tareas, y cómo se implementarán los mecanismos de seguridad.

Seguridad

Como se ha comentado, los datos utilizados en la herramienta son públicos por lo que no se requiere un control de los usuarios. Por otro lado se debe impedir que los usuarios modifiquen los datos de las terminologías almacenados. Por ello se establece el siguiente requisito:

- **REQ#17:** La API REST proporcionada a los usuarios únicamente permitirá la obtención de los datos relevantes a través de consultas GET. No se permitirá el acceso a la información mediante otras vias de acceso.

Fiabilidad

El sistema encuentra un alto grado de fiabilidad dada la naturaleza no dinámica de los datos que maneja. Básicamente, el sistema deberá asegurarse de la veracidad de los datos mostrados y asegurar un tiempo online la mayor parte del tiempo

- **REQ#18:** El sistema deberá asegurarse que la información mostrada corresponde con el concepto buscado.
- **REQ#19:** El sistema deberá relanzarse ante la aparición de un error crítico que provoque su apagado.

Mantenimiento

El sistema no deberá tener ningún mantenimiento a parte de asegurarse de que se disponga de una conexión a internet adecuada y un sistema capad de soportar la herramienta.

También se deberá realizar un mantenimiento ante posibles cambios de APIs y tecnologías utilizadas y principalmente se deberá volver a obtener un archivo OWL con la estructura utilizada ante la aparición de nuevas versiones de los datos de las terminologías.

Portabilidad

El sistema será totalmente portátil dada la naturaleza de las tecnologías Node.js y Servlets utilizadas. El sistema no soporta una plataforma multinodo sino que permite múltiples usuarios a través del mismo acceso.

4.2 Modelo de Casos de uso

Los casos de uso descritos en esta sección se han obtenido a partir de los requisitos de la sección previa y ayudan a determinar las situaciones más comunes de utilización del producto.

4.2.1 Actores

Los actores principales relacionados con los casos de uso son:

- **Usuarios:** Persona con un conocimiento básico de conceptos médicos y con un manejo a nivel de usuario de internet. Este agente será el principal interactuador con el sistema y el que iniciará todas las interacciones.
- **Sistema de consultas:** Sistema que se encarga de realizar una búsqueda de los datos de los conceptos y un análisis de los mismos para permitir una comprensión más sencilla por parte de los usuarios.

También encontramos un agente auxiliar que interactúa con el sistema de consultas a través de la interfaz REST, cualquier sistema externo puede hacer uso de esa API.

4.2.2 Diagrama de casos de uso

El diagrama de la figura 4.1 demuestra los principales casos de uso útiles que se pueden realizar con la herramienta desarrollada.

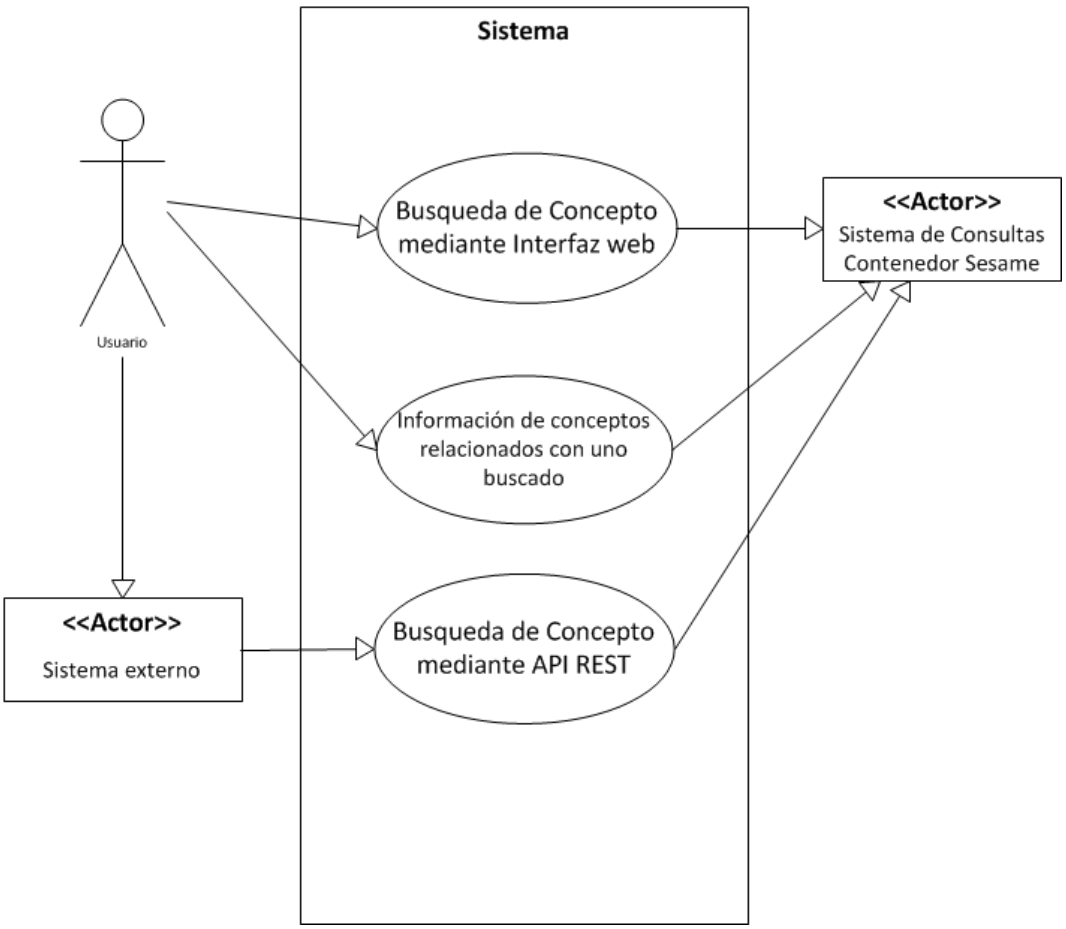

Figura 4.1: Diagrama Casos de Uso

4.2.3 Casos de uso

En esta sección describiremos con más detalle los casos de uso principales de la herramienta.

4.2.3.1 Caso 1: Búsqueda de concepto mediante la interfaz web

- **Actores:** Usuario, Sistema de consultas

- **Propósito:** Obtener información de un concepto médico dado su nombre o código asociado dentro del vocabulario.
- **Visión general:** El sistema muestra la información relevante del concepto al usuario a través de la interfaz web.
- **Importancia:** Primario, necesario
- **Precondiciones:** Conocer nombre o código del concepto o parte del nombre.
- **Referencias:** REQ#01, REQ#06, REQ#07, REQ#08, REQ#09, REQ#10, REQ#12, REQ#13, REQ#15
- **Escenario principal:**

Usuario	Sistema de consultas
1. Usuario introduce código del concepto médico del que desea buscar información	
	2. El sistema busca el concepto en el contenedor Sesame
	3. El sistema obtiene la forma normal a partir de los Servlets asociados
	4. El sistema analiza la información y obtiene una representación en JSON
5. El usuario recibe el JSON y visualiza la información en la aplicación web	

- **Escenarios alternativos:**
  En este caso todos los escenarios alternativos tienen que ver con la forma de realizar la búsqueda del concepto.

Usuario	Sistema de consultas
1a1. Usuario introduce nombre completo del concepto médico del que desea buscar información 1a2. Saltamos a paso 2.	
1b1. Usuario introduce parte del nombre del concepto médico	
	1b2. Sistema obtiene una lista de posibles conceptos que coinciden con el texto introducido.
1b3. El usuario recibe la lista de sugerencias de conceptos y selecciona uno.	
1b4. Saltamos a paso 1a1.	

4.2.3.2 Caso 2: Información de conceptos relacionados con uno buscado

- **Actores:** Usuario, Sistema de consultas
- **Propósito:** Obtener información de un concepto relacionado con uno ya buscado
- **Visión general:** El sistema realiza la navegación desde el concepto ya buscado hasta el concepto relacionado.
- **Importancia:** Primario, necesario
- **Precondiciones:** Haber realizado una búsqueda previa de un concepto y visualizar dicho concepto a través de la herramienta web.
- **Referencias:** REQ#02, REQ#03, REQ#05, REQ#08, REQ#09, REQ#10, REQ#13, REQ#15
- **Escenario principal:**

Usuario	Sistema de consultas
1. Usuario hace click en alguno de los conceptos que aparecen en las relaciones de un concepto previamente buscado.	
	2. El sistema busca el concepto relacionado en el contenedor Sesame a partir de su código dentro de la terminología.
	3. El sistema obtiene la forma normal a partir de los Servlets asociados
	4. El sistema analiza la información y obtiene una representación en JSON
5. El usuario recibe el JSON y visualiza la información en la aplicación web pudiendo volver a visualizar el concepto original con un simple click.	

4.2.3.3 Caso 3: Búsqueda de concepto mediante la API REST

- **Actores:** Usuario, Sistema de consultas
- **Propósito:** Obtener información de un concepto médico dado su nombre o código asociado dentro del vocabulario utilizando una herramienta externa.
- **Visión general:** El sistema transmite la información relevante del concepto a la herramienta de terceros a través de la API REST en formato JSON.
- **Importancia:** Secundario, opcional
- **Precondiciones:** Conocer código del concepto a buscar dentro de la terminología a la que pertenece.
- **Referencias:** REQ#01, REQ#09, REQ#10, REQ#11, REQ#12, REQ#14, REQ#15, REQ#16, REQ#17
- **Escenario principal:**

Usuario	Herramienta Externa	Sistema de consultas
1. Usuario utiliza herramienta externa que hace uso de información de conceptos médicos		
	2. La herramienta externa realiza una petición a la API sobre la información de los conceptos médicos que necesita.	
		3. El sistema busca los datos del concepto en el contenedor Sesame a partir de su código dentro de la terminología.
		4. Si necesita esa información, el sistema obtiene la forma normal a partir de los Servlets asociados
		5. El sistema analiza la información y obtiene una representación en JSON
	6. La herramienta externa recibe el JSON y lo utiliza para realizar sus funciones	

4.3 Implementación

4.3.1 Modificaciones a la versión oficial de SNOMED-CT en OWL

Como mencionamos en la sección 3.6.1, la distribución oficial de SNOMED-CT viene acompañada de un script PERL para la generación de un fichero OWL conteniendo esta terminología. Si bien esta información contiene la información del vocabulario, se han realizado diversas modificaciones para ampliar dicha información con otros datos obtenidos en el presente trabajo. Esta información es:

- **Duplicación de información de parentesco:** Esta información viene incluida en el OWL original pero se encuentra en un formato que dificulta la accesibilidad. Para facilitar el acceso a esta información se duplica de forma más accesible. Podemos observar un ejemplo de este cambio en la figura 4.2.

```
1  <owl:Class rdf:about="SCT_287299000">
2     <rdfs:label>Chest wall adhesions freed</rdfs:label>
3     <skos:altLabel>Chest wall adhesions freed</skos:altLabel>
4     <prv:containedBy rdf:resource="http://test.org#Procedure_code"/>
5     <!-- La linea 6 es una duplicacion de la linea 9 original -->
6     <rdfs:subClassOf rdf:resource="SCT_443938003"/>
7     <owl:equivalentClass><owl:Class>
8     <owl:intersectionOf rdf:parseType="Collection">
9         <owl:Class rdf:about="SCT_443938003"/>
10        <owl:Restriction>
11        ...
```

Figura 4.2: Ejemplo de duplicación de información de parentesco

- **Uso de datos inferenciados:** La distribución de SNOMED-CT incluye dos versiones de la información una inferenciada y otra sin inferenciar. La información de uso del script indica que se debe utilizar la versión no inferenciada pero dado que eso provocaría procesamientos adicionales posteriores y el formato en el que están ambas versiones es el mismo (no se requería ningún cambio), se decidió utilizar la versión inferenciada que automáticamente adjunta cada concepto con las relaciones que contiene en lugar de tener que calcularlas a partir de relaciones que se hereden de conceptos padres e hijos.

- **Inclusión de información del Term Info:** Para poder acceder a la información proporcionada por el proceso del term info y conocer con que ámbito se pueden relacionar cada concepto, esta información es introducida en el OWL modificando el script Perl. Para ello se hace uso del vocabulario Provenance [39] que incluye clases y etiquetas adicionales a las utilizadas en OWL y RDF. Finalmente la sintaxis de esta información sería:

```
1  <prv:containedBy rdf:resource="TERMINFO_CLASS"/>
```

  TERMINFO_CLASS se obtendría a partir del módulo correspondiente realizado en los proyectos INTEGRATE y EURECA, se almacenaría en un fichero determinando esta clase por cada concepto de la terminología y este fichero es utilizado por el script para incluir esta información.

- **Inclusión de sinónimos:** En SNOMED-CT existen una cantidad de sinónimos asociados a cada concepto. Dado que la herramienta permite realizar búsquedas por el nombre de estos conceptos, esta información debe estar incluida en el OWL. Dado que la versión original solo incluye el nombre completamente definido (Fully-defined), se ha realizado una modificación a el script Perl para incluir dicha información utilizando SKOS (Simple Knowledge Organization System) [40], uno de los estándares creados sobre RDF para ampliar su capacidad de definición de información. La sintaxis resultante es:

```
1 <skos:altLabel>SINONIMO</skos:altLabel>
```

  Donde SINONIMO es obtenido a partir de uno de los ficheros de información adicionales que se incluyen en la distribución de SNOMED-CT. La sintaxis de "skos:altLabel" se refiere a Título alternativo o sinónimo.

- **Inclusión de vocabularios LOINC y HGNC:** Se decidió ampliar el OWL generado por el Script Perl de SNOMED-CT para incluir información de los vocabularios LOINC y HGNC. Esto es debido al uso adicional que se realiza de este archivo en los proyectos INTEGRATE y EURECA los cuales utilizan información de estos vocabularios. LOINC y HGNC no presentan la misma estructura que la que presentan los conceptos de SNOMED-CT. Debido a esto no se ha implementado la utilización de estos conceptos en la herramienta pese a que unos ajustes sencillos permitirían hacerlo. Ejemplos de la estructura en el OWL de los conceptos de estas dos terminologías se representan en las figuras 4.3 para LOINC y 4.4 para HGNC.

```
<owl:Class rdf:about="http://www.loinc.org/31781-8">
  <rdfs:label>Classical swine fever virus Ag</rdfs:label>
  <prv:containedBy rdf:resource="test.org#Observation_code"/>
  <rdfs:subClassOf rdf:resource="SCT_000000051"/>
  <loinc:Property>ACnc</loinc:Property>
  <loinc:TimeAspct>Pt</loinc:TimeAspct>
  <loinc:System>XXX</loinc:System>
  <loinc:ScaleTyp>Ord</loinc:ScaleTyp>
  <loinc:MethodTyp></loinc:MethodTyp>
  <loinc:Class>MICRO</loinc:Class>
  <loinc:ClassType>Lab</loinc:ClassType>
  <loinc:Shortname>CSFV Ag XXX Ql</loinc:Shortname>
  <loinc:OrderObs>Both</loinc:OrderObs>
  <loinc:LongCommonName>Classical swine fever virus Ag [Presence]
     in Unspecified specimen</loinc:LongCommonName>
</owl:Class>
```

Figura 4.3: Ejemplo de concepto de LOINC en el OWL

```
<owl:Class rdf:about="http://www.hgnc.org/RNA5SP195">
  <rdfs:label xml:lang="en">RNA, 5S ribosomal pseudogene 195</
     rdfs:label>
  <prv:containedBy rdf:resource="http://test.org#
     Observation_targetSiteCode"/>
  <prv:containedBy rdf:resource="http://test.org#
     Procedure_targetSiteCode"/>
  <rdfs:subClassOf rdf:resource="SCT_000000061"/>
</owl:Class>
```

Figura 4.4: Ejemplo de concepto de HGNC en el OWL

4.3.2 Flujo de datos

En el diagrama de la figura 4.5 podemos observar los datos que se desplazan a través de los principales procesos del sistema.

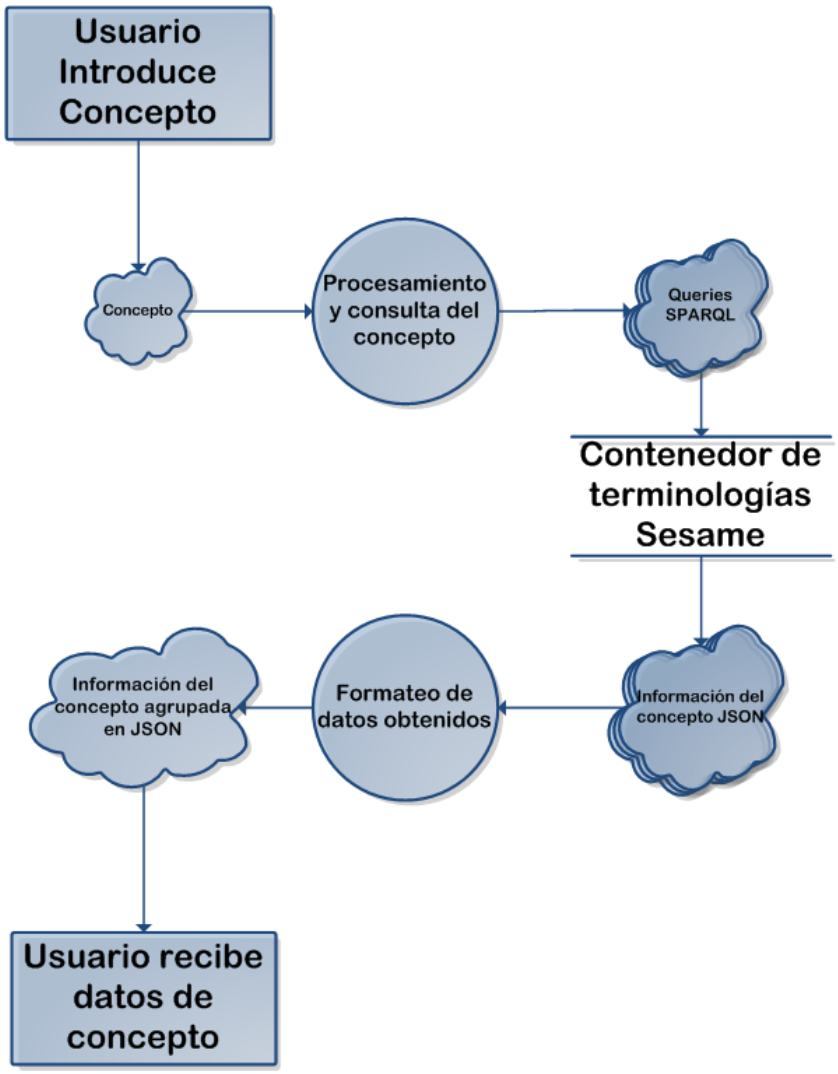

Figura 4.5: Diagrama de Flujo de Datos

Siguiendo el diagrama 4.5 observamos que el usuario introduce un concepto del cual desea obtener toda o parte de la información. Este concepto se procesa (en el servicio de Node.js) y se encarga de realizar una serie de consultas al contenedor Sesame obteniendo la información relevante. Cada una de estas consultas devuel-

ve la información solicitada en JSON. Esta información es tratada para agruparla y proporcionarle el formato adecuado que se espera en la salida. Finalmente esta información es devuelta al usuario en formato JSON. Este diagrama de flujo sería exactamente el mismo en caso de realizarse a través de la API REST, si bien el agente que interactúa con el servicio no sería el usuario sino una herramienta externa. Este diagrama de flujo de datos se refiere a cualquier petición de datos realizada al servicio incluyendo peticiones de sugerencias de conceptos.

Cabe tener en cuenta que la consulta y procesamiento de la forma normal de los conceptos se realiza en un proceso a parte del cual hace uso el propio servicio alojado en un servidor en Node.js.

Si bien cabe la posibilidad de que el contenedor Sesame se encuentre en una máquina diferente respecto a la máquina donde se ejecuta el servicio, es recomendable que este sea local para agilizar las consultas realizadas. También es útil por cuestiones de seguridad al permitirnos cerrar el acceso al contenedor directamente excepto en accesos locales. Como se ha comentado, esto es útil para permitir a los usuarios realizar únicamente las operaciones permitidas y evitar que se modifiquen los datos almacenados.

4.3.3 Diagrama de secuencia

En el diagrama de la figura 4.6 observamos la interacción entre los diferentes componentes que permiten realizar las operaciones. En concreto este diagrama de secuencia corresponde con el caso de uso 4.2.3.1 incluyendo las ampliaciones que se pueden realizar sobre las operaciones de dicho caso de uso.

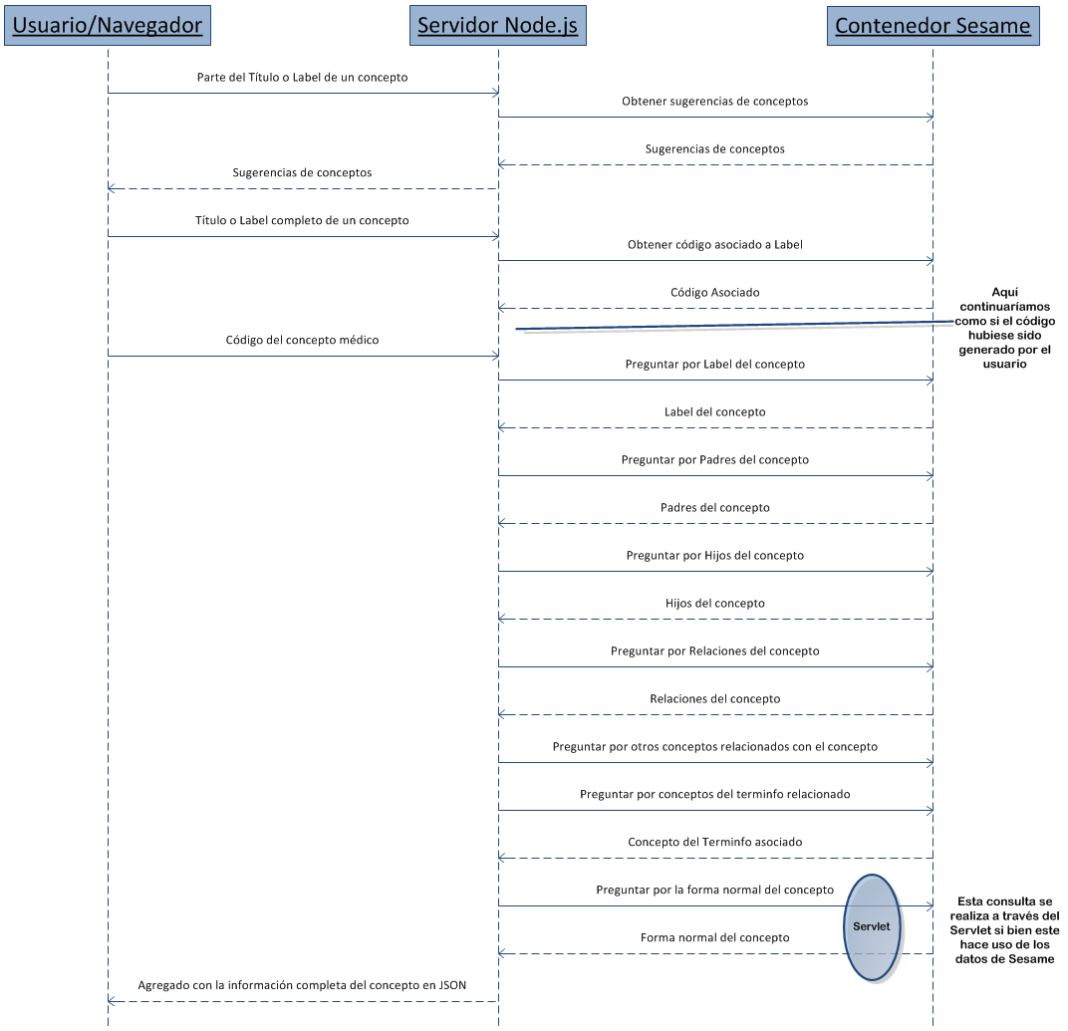

Figura 4.6: Diagrama de Secuencia

4.3.4 Ficheros del proyecto

En el diagrama de la figura 4.7 observamos la estructura que siguen los ficheros que conforman el cuerpo del servidor y el cliente de la herramienta.

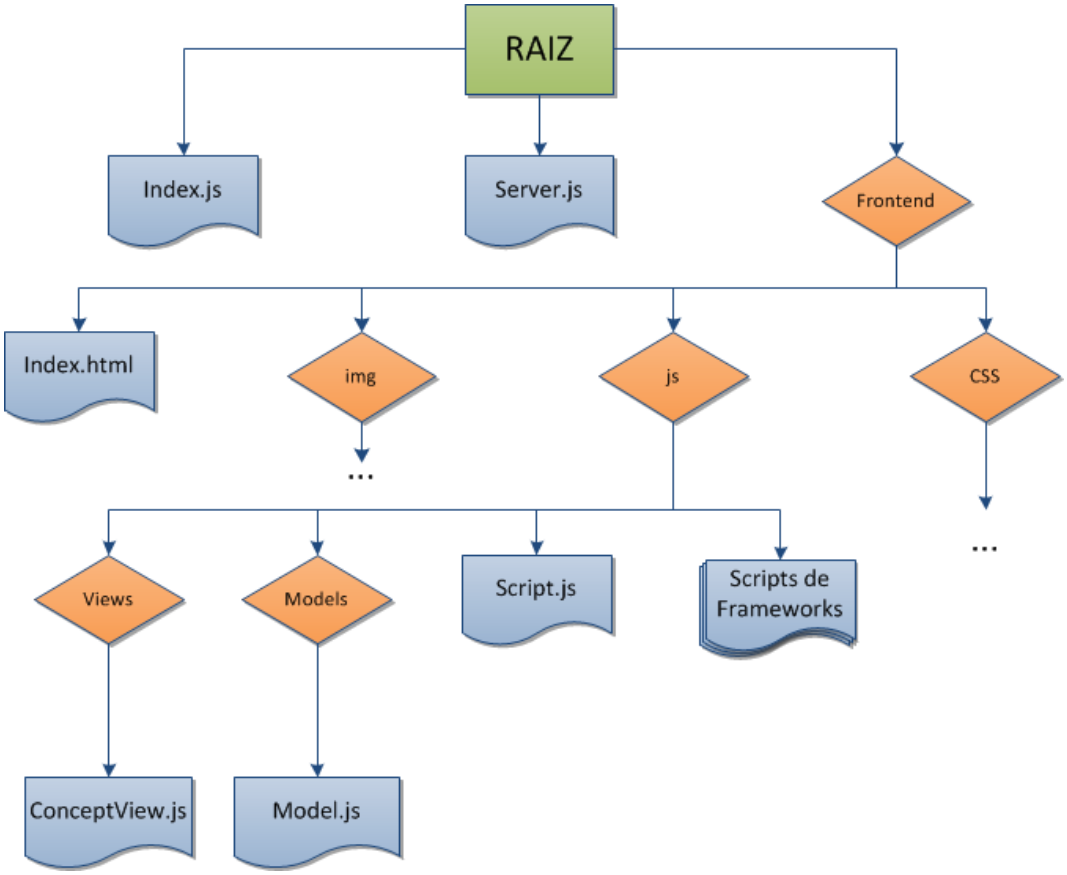

Figura 4.7: Diagrama de Ficheros

A continuación incluimos una pequeña descripción de la utilizad de los principales ficheros mostrados en este diagrama.

- **Index.js:** El fichero index.js será utilizado como el fichero que iniciará el servidor Node.js indicando puerto de ejecución y otras variables globales a utilizar por el servidor.
- **Server.js:** El fichero Server.js contendrá toda la API REST creada por la herramienta. También realizará las operaciones necesarias para el acceso al

contenedor Sesame y el procesamiento y formato de esa información. También se encargará de transmitir la página principal del frontend a los usuarios.

- **Index.html:** El fichero principal del servidor transmitido a los usuarios a través del servidor en Node.js. Contendrá la estructura de la aplicación a rellenar con posteriores interacciones entre el servidor y el cliente.
- **Script.js:** Este fichero contendrá el código Javascript del lado del cliente que se encargará de actualizar la página HTML dinámicamente de acuerdo con las operaciones realizadas.
- **Scripts de frameworks:** Aquí se encuentran todos los frameworks utilizados en la elaboración de la herramienta en lo que se refiere a las funcionalidades en el lado del cliente (o Frontend). Esto incluye librerías como Backbone.js, Jquery, Underscore.js, Raphael.js...
- **ConceptView.js:** Fichero que contendrá la vista de Backbone de un concepto genérico. Una de estas vistas es generada por cada concepto visible en la herramienta. Este fichero se encarga de actualizar la parte gráfica de cada concepto según cualquier cambio en el modelo.
- **Model.js:** El fichero Model.js contiene el modelo de Backbone y contendrá la estructura de los conceptos médicos. También establece la url del servicio REST en la que deberá de consultar para obtener los datos de los conceptos.

Además de estos ficheros existen algunos más en las carpetas "img" que contendrá las imágenes utilizadas en la herramienta y "css" que contendrá reglas de aspecto a seguir por la herramienta. Como ambas no afectan a la funcionalidad se ha decidido no incluir una descripción de las mismas.

4.3.5 Modelo de datos

Como se ha comentado en otras secciones, todos los datos utilizados se encontrarán en formato JSON con la única excepción de los datos de las terminologías que se encontrarán en formato RDF y almacenadas en el contenedor Sesame. La estructura JSON creada para la transmisión de los datos se muestra en la figura 4.8. En esta estructura observamos cómo cada dato viene representado siguiendo el formato clave-valor de JSON siendo cada par un dato relevante del concepto.

A continuación definiremos los diferentes campos de este modelo de datos.

```
{
  "code": "71620000",
  "title": "Fracture of femur",
  "parents": [{
      "code": "7523003",
      "title": "Injury of thigh (disorder)"
    },...],
  "children": [{
      "code": "206213006",
      "title": "Fracture of femur due to birth trauma"
    },...],
  "relationships": [{
      "relationship": "Associated morphology (attribute)",
      "relationshipCode": "116676008",
      "title": "Fracture (morphologic abnormality)",
      "code": "72704001"
    },...],
  "related": [{
      "relationship": "After",
      "relationshipCode": "255234002",
      "title": "Late effect of fracture of neck of femur",
      "code": "17194005"
    },...],
  "terminfo": "http://test.org#Observation_code",
  "shortnormalform": "NORMALFORM"
}
```

Figura 4.8: Ejemplo JSON

- **code:** Este campo contendrá el código único del concepto dentro del vocabulario correspondiente.
- **title:** Este campo contendrá el nombre preferido del concepto dentro del vocabulario correspondiente.

- **parents:** Este campo contendrá un array con información de los padres directos del concepto. Esta información será el código y el título de cada concepto padre.
- **children:** Este campo contendrá un array con información de los hijos directos del concepto. Esta información será el código y el título de cada concepto hijo.
- **relationships:** Este campo contendrá un array con información de las relaciones definitorias del concepto. Esta información será el código y el título del concepto objeto de la relación y el código y el título de la relación concreta que los asocia.
- **related:** Este campo contendrá un array con información de los conceptos relacionados inversamente con el concepto en cuestión. Esta información será el código y el título del concepto origen de la relación y el código y el título de la relación concreta que los asocia.
- **terminfo:** Este campo contendrá la clase del terminfo asociada al concepto.
- **shortnormalform:** Este campo contendrá la forma normal del concepto en un formato pseudo-JSON.

La información mostrada en la figura 4.8 es una parte de la información obtenida al utilizar el servicio REST que permite obtener toda la información de un concepto con una única llamada. Cada sección de los datos explicados anteriormente puede ser obtenida de forma separada con su respectiva llamada al servicio concreto para obtenerla.

4.3.6 Estructura del Servlet utilizado para la obtención de la forma normal

El servlet creado para la obtención de la forma normal presenta cuatro clases en dos paquetes diferentes cuya definición completa puede observarse en el diagrama 4.9.

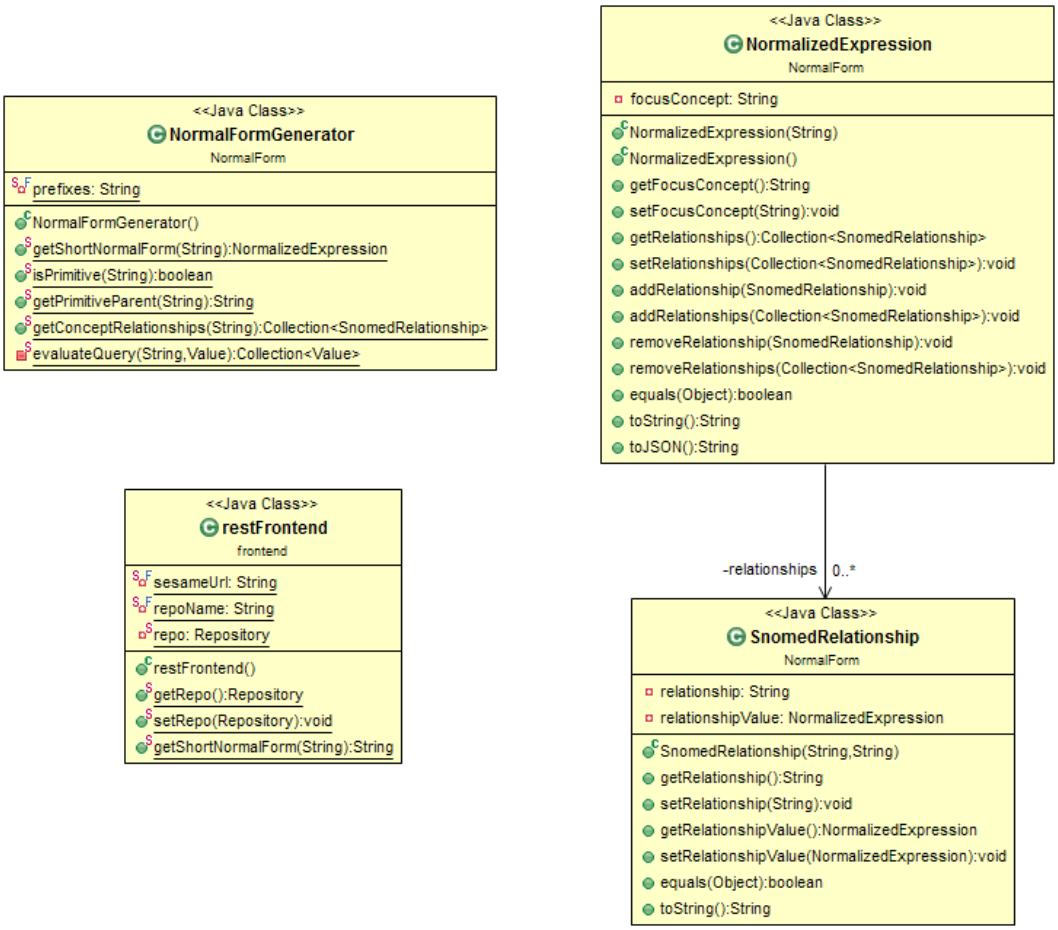

Figura 4.9: Diagrama de Clases del Servlet

La clase "NormalFormGenerator" incorpora el método para la obtención de la forma normal y una serie de métodos auxiliares para ello. La forma normal es devuelta siguiendo la clase "NormalizedExpression" que presenta un focus concept y un array de "SnomedRelationship" una clase que presenta un par relación-valor donde el valor es a su vez otra "NormalizedExpression". Finalmente tenemos la clase "restFrontend" que se encargará de ofrecer el método de obtención de la forma normal en

la interfaz del Servlet. Centrándonos en los métodos y atributos concretos de cada clase tenemos:

NormalFormGenerator

- **getShortNormalForm:** Obtiene la forma normal corta de un concepto dado (como un String en su forma de código de SNOMED-CT)
- **isPrimitive:** Indica si el código introducido corresponde a un concepto primitivo.
- **getPrimitiveParent:** Recorre los padres del concepto dado y devuelve el código del padre primitivo más cercano.
- **getConceptRelationships:** Devuelve las relaciones definitorias de un concepto dado.

Como atributos tenemos únicamente los prefijos a utilizar en las queries de SPARQL.

NormalizedExpression

- **addRelationship:** Existen 2 métodos con este nombre que añaden una o varias relaciones al conjunto de relaciones definitorias de la forma normal resultante.
- **removeRelationship:** Existen 2 métodos con este nombre que eliminan una o varias relaciones del conjunto de relaciones definitorias de la forma normal resultante.
- **toJSON:** Obtiene una representación en pseudo-JSON de la forma normal.

Además de estos métodos existen los métodos equals (para comparar entre este objeto y uno dado), toString (para obtener una representación en formato de String) y una serie de getters y setters para los atributos de la clase que son el focusConcept y las relationships, una colección de SnomedRelationships representadas en el diagrama con una relación entre ambas clases con una multiplicidad de 0 a n.

SnomedRelationship

Esta clase no presenta métodos de importancia sino que se trata más concretamente de un contenedor de información para la relación y el valor de la relación (que es a su vez una NormalizedExpression). Los métodos son únicamente getters y setters, equals y el método toString.

restFrontend

Esta clase actúa de interfaz del servicio REST del Servlet. El único método disponible en esa interfaz es el método getShortNormalForm. Como atributos tenemos la URL del contenedor Sesame donde conectarnos, el nombre del repositorio y el repositorio que se crea al iniciar el Servlet.

4.3.7 Especificación de API REST

A continuación se definen los servicios pertenecientes a la API REST creada para su uso en la interfaz o mediante herramientas externas.

- **Servicio para obtención de label de un concepto**

URI del recurso	/snomed/concept/{conceptID}/label
Operacion CRUD	GET
Definición de operación	Obtiene el label o título del concepto con código "conceptID" dentro de la terminología SNOMED
Definición de atributos	conceptID — Código del concepto
Posibles valores de retorno	200 OK
Ejemplo resultado operación	`{label: "Fracture of femur"}`

- **Servicio para obtención de padres de un concepto**

URI del recurso	/snomed/concept/{conceptID}/parents
Operacion CRUD	GET
Definición de operación	Obtiene los padres del concepto con código "conceptID" dentro de la terminología SNOMED
Definición de atributos	conceptID — Código del concepto
Posibles valores de retorno	200 OK
Ejemplo resultado operación	

```
{
  "parents": [{
      "code": "7523003",
      "title": "Injury of thigh (
         disorder)"
    },...]
}
```

- **Servicio para obtención de hijos de un concepto**

URI del recurso	/snomed/concept/{conceptID}/children
Operacion CRUD	GET
Definición de operación	Obtiene los hijos del concepto con código "conceptID" dentro de la terminología SNOMED
Definición de atributos	conceptID — Código del concepto
Posibles valores de retorno	200 OK
Ejemplo resultado operación	

```
{
  "children": [{
      "code": "206213006",
      "title": "Fracture of femur due to
          birth trauma"
    },...]
}
```

- **Servicio para obtención de relaciones de un concepto**

URI del recurso	/snomed/concept/{conceptID}/relationships
Operacion CRUD	GET
Definición de operación	Obtiene las relaciones del concepto con código "conceptID" dentro de la terminología SNOMED
Definición de atributos	conceptID — Código del concepto
Posibles valores de retorno	200 OK
Ejemplo resultado operación	

```
{
  "relationships": [{
      "relationship": "Associated
         morphology (attribute)",
      "relationshipCode": "116676008",
      "title": "Fracture",
      "code": "72704001"
    },...]
}
```

- **Servicio para obtención de conceptos relacionados con un concepto**

URI del recurso	/snomed/concept/{conceptID}/related
Operacion CRUD	GET
Definición de operación	Obtiene los conceptos relacionados con el concepto con código "conceptID" dentro de SNOMED
Definición de atributos	conceptID — Código del concepto
Posibles valores de retorno	200 OK
Ejemplo resultado operación	

```
{
  "related": [{
      "relationship": "After",
      "relationshipCode": "255234002",
      "title": "Late effect ...",
      "code": "17194005"
    },...]
}
```

- **Servicio para obtención del concepto del term info de un concepto**

URI del recurso	/snomed/concept/{conceptID}/terminfo
Operacion CRUD	GET
Definición de operación	Obtiene el concepto del term info asociado con el concepto con código "conceptID" dentro de la terminología SNOMED
Definición de atributos	conceptID — Código del concepto
Posibles valores de retorno	200 OK
Ejemplo resultado operación	`{ "terminfo": "http://test.org#Observation_code" }`

- **Servicio para obtención de la forma normal corta de un concepto**

URI del recurso	/snomed/concept/{conceptID}/terminfo
Operacion CRUD	GET
Definición de operación	Obtiene la forma normal corta del concepto con código "conceptID" dentro de la terminología "SNOMED"
Definición de atributos	conceptID — Código del concepto
Posibles valores de retorno	200 OK
Ejemplo resultado operación	`{ "shortnormalform": "NORMALFORM" }`

- **Servicio para obtención de toda la información de un concepto**

URI del recurso	/{vocID}/concept/{conceptID}
Operacion CRUD	GET
Definición de operación	Obtiene toda la información del concepto con código "conceptID" dentro de la terminología "vocID". Hace uso del resto de servicios
Definición de atributos	conceptID — Código del concepto
Posibles valores de retorno	200 OK
Ejemplo resultado operación	Vease figura 4.8.

- **Servicio para obtención de sugerencias de un concepto dado un label**

URI del recurso	/{vocID}/suggestions/{text}
Operacion CRUD	GET
Definición de operación	Obtiene un listado de conceptos cuyo título contiene todo o parte de "text"
Definición de atributos	vocID — Código del vocabulario, text — Texto a buscar
Posibles valores de retorno	200 OK
Ejemplo resultado operación	

```
1 {
2   "suggestions": [{
3       "code": "206213006",
4       "title": "Fracture of femur due to
            birth trauma"
5     },...]
6 }
```

- **Servicio para obtención de sugerencias de un concepto dado un label filtrando por el concepto del term info**

URI del recurso	/{vocID}/suggestions/{text}/{terminfo}
Operacion CRUD	GET
Definición de operación	Obtiene un listado de conceptos cuyo título contiene todo o parte de “text” y el código asociado del term info es “terminfo”
Definición de atributos	vocID — Código del vocabulario, text — Texto a buscar, terminfo — concepto del terminfo por el que filtrar
Posibles valores de retorno	200 OK
Ejemplo resultado operación	Vease ejemplo anterior.

Capítulo 5

PRUEBAS DE LA HERRAMIENTA

5.1 Pruebas

En esta sección detallaremos pruebas de rendimiento, fiabilidad e interacción realizadas a la herramienta y al servicio REST que utiliza. Esto es una parte imprescindible del desarrollo que se realizó iterativamente con cada cambio significativo del código fuente.

5.1.1 Pruebas de rendimiento

Para comprobar el rendimiento del servicio, hemos utilizado un cliente que realiza una serie de consultas consecutivas al servicio y calculamos un valor de tiempo medio de las peticiones, este valor para consultas paralelas aumenta ligeramente conforme al número de operaciones simultáneas. Por otro lado el tiempo necesario para realizar una única consulta simultánea se sitúa alrededor de 1 segundo. Podemos ver una evolución de los tiempos en la figura 5.1.

Estas pruebas se realizaron en un ordenador de sobremesa con las siguientes características:

- **Sistema Operativo:** Windows 7 Professional
- **Procesador:** Intel Core 2 Quad Q6600 2.40GHz
- **Memoria:** 4GB de Memoria RAM
- **Disco duro:** 300GB a 7200rpm

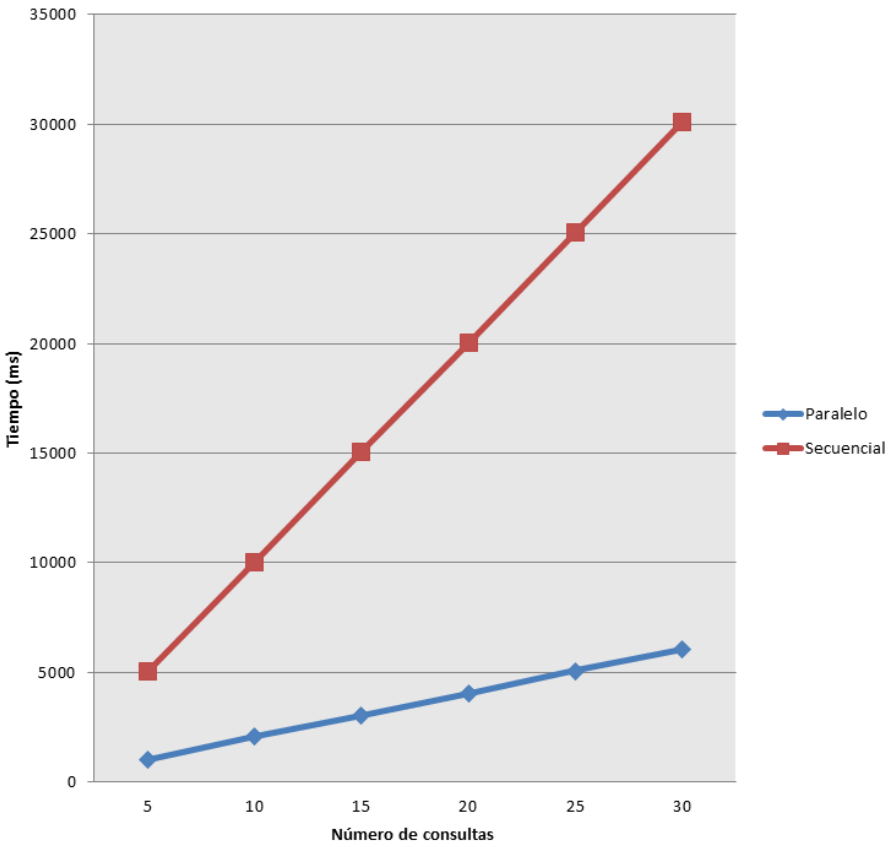

Figura 5.1: Comparación de tiempos para realizar operaciones paralela y secuencialmente

El principal inconveniente a la hora de realizar las pruebas fue la memoria que era ocupada mayormente por el contenedor Sesame con un consumo de cerca de 3GB de memoria. Esto provocaba una ralentización en los servicios ejecutados.

Los resultados mostrados en la figura 5.1 nos permiten observar que una consulta individual se puede realizar en un tiempo medio de 1 seg. si bien las consultas paralelas no mantienen esa regla aumentando ligeramente con el número de consultas simultáneas. Esto nos permite estimar el funcionamiento ante un número elevado de usuarios. Este aumento de tiempo en las consultas paralelas es probable se vea afectado por el rendimiento de la máquina donde se ejecuta el servidor y su tendencia a aumentar el tiempo podría ser menor con una máquina con características superiores.

5.1.2 Pruebas de fiabilidad

Para comprobar la fiabilidad de la forma más eficiente posible se han contrastado los resultados obtenidos por el servicio con los resultados obtenidos con otros visores de información de SNOMED-CT como UMLS o Bioportal. En las pruebas realizadas no se han encontrado contradicciones entre los valores mostrados por la herramienta y los valores de estos otros portales. En las figuras 5.2 y 5.3 se pueden observar unas capturas de pantalla de los resultados mostrados en UMLS y Bioportal respectivamente.

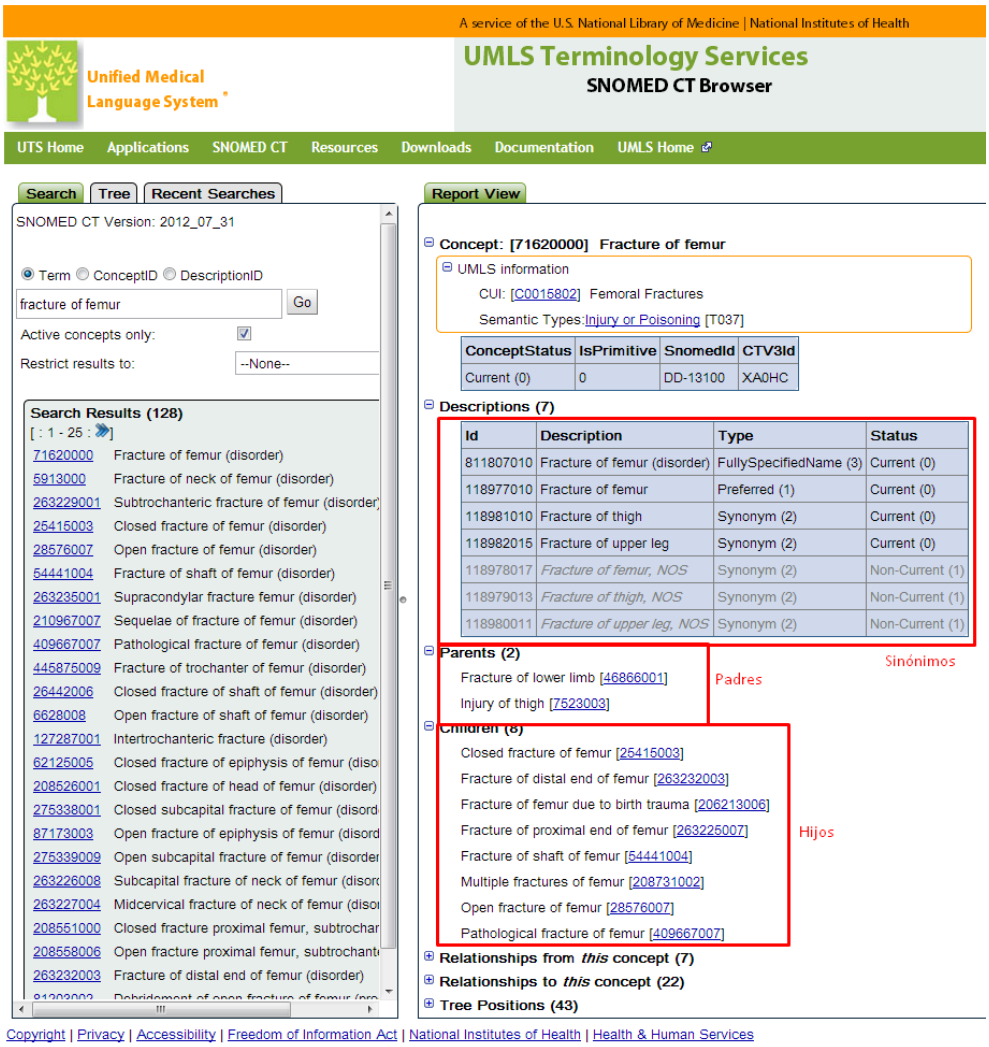

Figura 5.2: UMLS Concepto Fracture of Femur - Padres e Hijos

Figura 5.3: Biportal Concepto Breast Cancer - Relaciones

Si bien la forma normal de los conceptos de SNOMED-CT es un dato que no aparece directamente en estos dos servicios, esta representación se puede obtener tras realizar un análisis de los datos que si estan mostrados en ellas. Por ello podemos confirmar a su vez la fiabilidad de la forma normal.

c

Tras hacer comprobaciones con estos ejemplos y con ejemplos de conceptos con diferentes estructuras y datos podemos confirmar la validez de los datos mostrados por la herramienta implementada en este proyecto.

5.2 Ejemplos de consulta de conceptos

A continuación expondremos algunas de las consultas que se han utilizado para comprobar el correcto funcionamiento de la herramienta. Se acompañan capturas de pantalla de los resultados de cada consulta en la herramienta web y los resultados en JSON del servicio REST que se utiliza.

- **Consulta los padres del concepto Fracture of femur**
  La consulta realizada por la herramienta se realiza a través del servicio web que permite obtener toda la información de un concepto (URL: localhost:8081/snomed/71620000). Por otro lado si utilizamos el servicio concreto para obtener los padres de un concepto (localhost:8081/snomed/71620000/parents) obtendríamos concretamente la información requerida en formato JSON que viene acompañado en la figura 5.4. Una captura de pantalla de la herramienta visualizando esta información viene acompañada en la figura 5.8.

```
{
 "parents": [
   {
     "code": "7523003",
     "title": "Injury of thigh (disorder)"
   },
   {
     "code": "46866001",
     "title": "Fracture of lower limb (disorder)"
   }
 ]
}
```

Figura 5.4: JSON padres de concepto Fracture of femur

- **Consulta los hijos del concepto 71620000 (Fracture of femur)**
  La consulta realizada por la herramienta se realiza a través del servicio web que permite obtener toda la información de un concepto (URL: localhost:8081/snomed/71620000). Por otro lado si utilizamos el servicio concreto para obtener los hijos de un concepto (localhost:8081/snomed/71620000/children) obtendríamos concretamente la información requerida en formato JSON que viene acompañado en la figura 5.5. Una captura de pantalla de la herramienta visualizando esta información viene acompañada en la figura 5.9.

```
{
 "children": [{
     "code": "206213006",
     "title": "Fracture of femur due to birth trauma (disorder)"
   },{
     "code": "208731002",
     "title": "Multiple fractures of femur (disorder)"
   },....
 ]}
```

Figura 5.5: JSON hijos de concepto Fracture of femur

- **Consulta la forma normal del concepto Epirubicin**
  La consulta realizada por la herramienta se realiza a través del servicio web que permite obtener toda la información de un concepto (URL: localhost:8081/snomed/417916005). Por otro lado si utilizamos el servicio concreto para obtener la forma normal de un concepto (localhost:8081/snomed/417916005/shortnormalform) obtendríamos concretamente la información requerida en formato JSON que viene acompañado en la figura 5.6. Una captura de pantalla de la herramienta visualizando esta información viene acompañada en la figura 5.11.

```
{ focusConcept: 417916005|Epirubicin (substance)}
```

Figura 5.6: JSON Forma normal de concepto Epirubicin

- **Consulta las relaciones del concepto Breast cancer**
  La consulta realizada por la herramienta se realiza a través del servicio web que permite obtener toda la información de un concepto (URL: localhost:8081/snomed/254837009). Por otro lado si utilizamos el servicio concreto para obtener las relaciones de un concepto (localhost:8081/snomed/254837009/relationships) obtendríamos concretamente la información requerida en formato JSON que viene acompañado en la figura 5.7. Una captura de pantalla de la herramienta visualizando esta información viene acompañada en la figura 5.10.

```
{
 "relationships": [{
     "relationship": "Finding site (attribute)",
     "relationshipCode": "363698007",
     "title": "Breast structure (body structure)",
     "code": "76752008"
   },{
     "relationship": "Associated morphology (attribute)",
     "relationshipCode": "116676008",
     "title": "Malignant neoplasm of primary, secondary, or
        uncertain origin (morphologic abnormality)",
     "code": "367651003"
  }]
}
```

Figura 5.7: JSON Relaciones de concepto Breast cancer

Todas los ejemplos de consultas anteriores se pueden observar en las capturas de pantalla de las figuras 5.8, 5.9, 5.10 y 5.11.

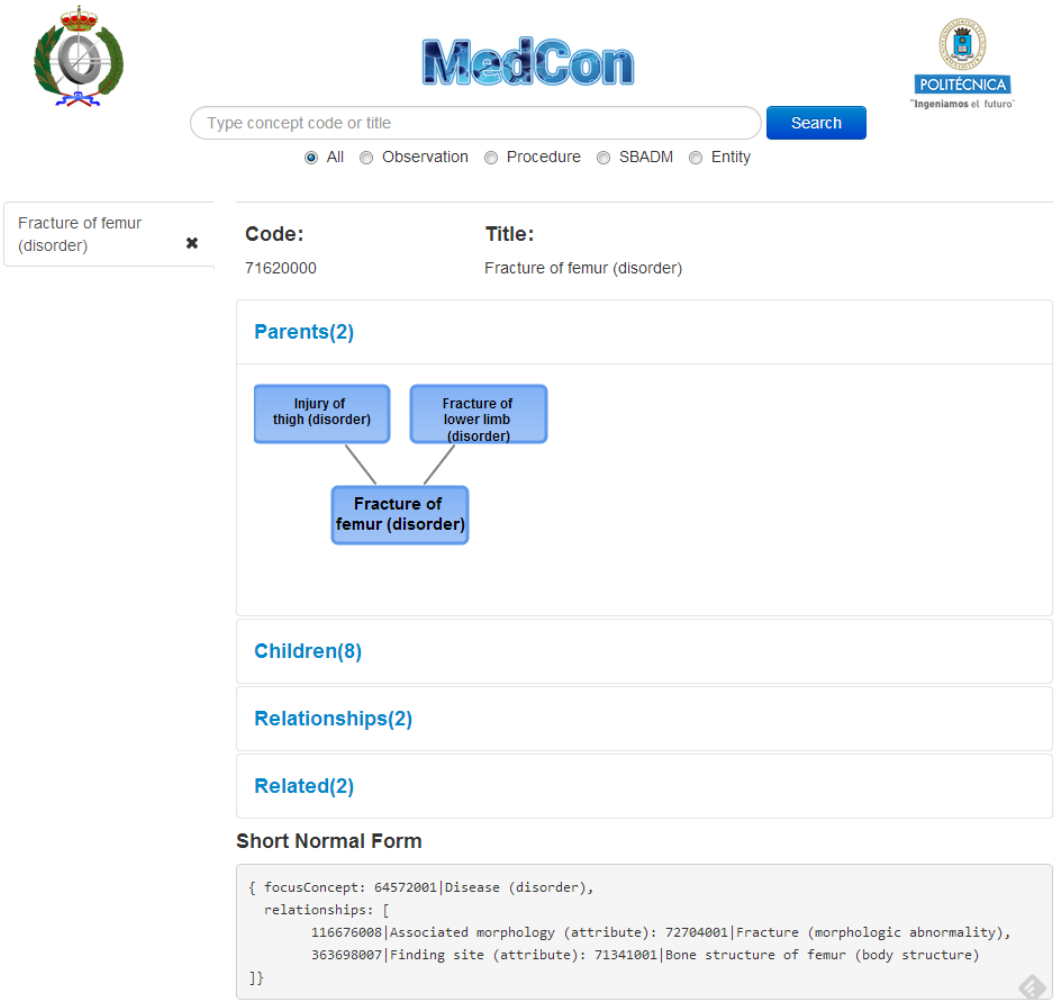

Figura 5.8: Captura de pantalla de la herramienta visualizando padres de Fracture of femur

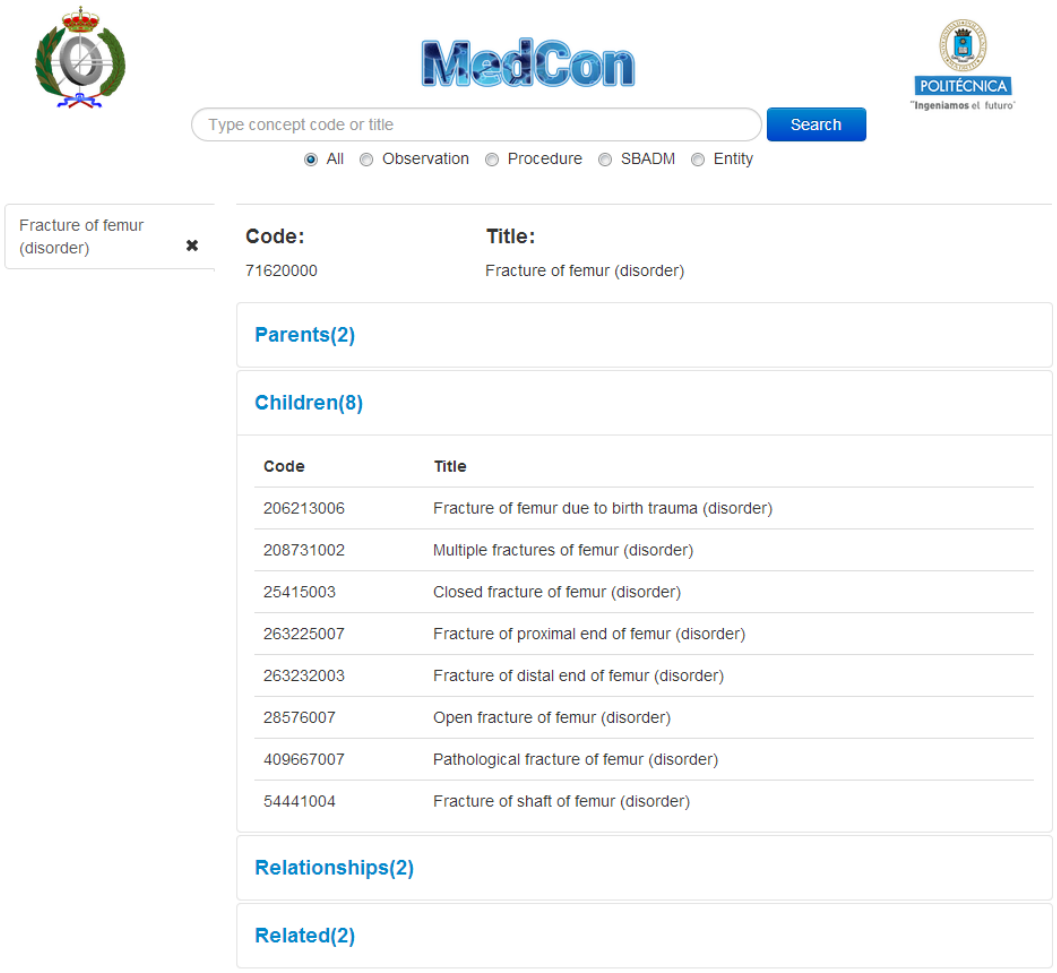

Figura 5.9: Captura de pantalla de la herramienta visualizando hijos de Fracture of femur

Figura 5.10: Captura de pantalla de la herramienta visualizando relaciones de Breast cancer

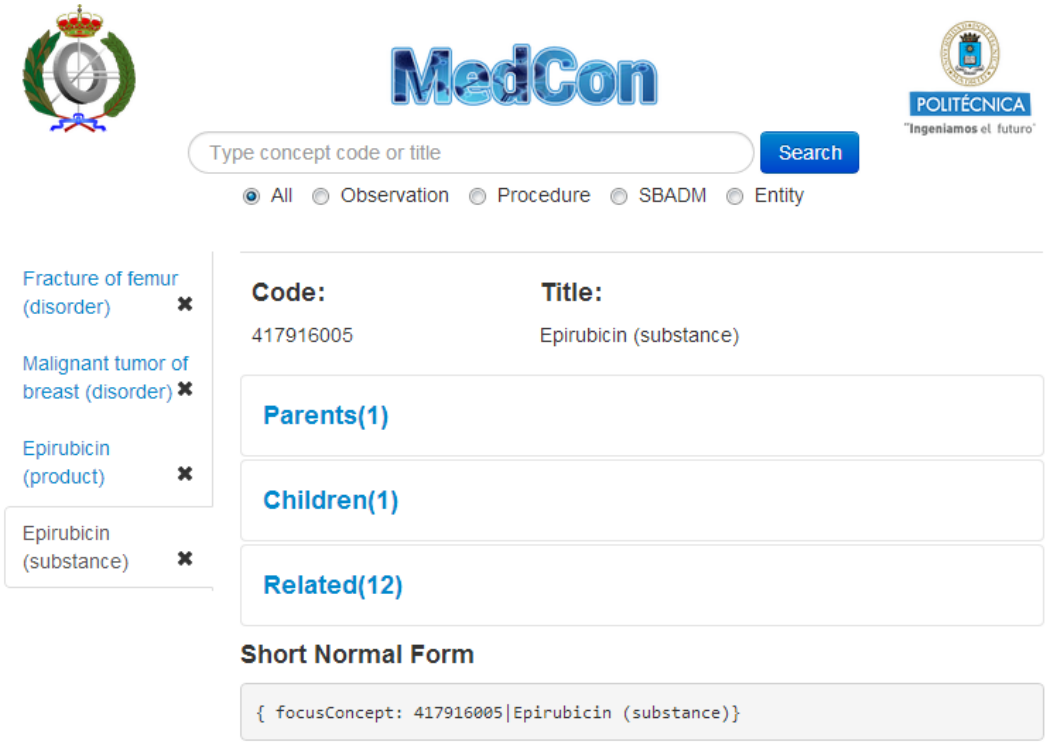

Figura 5.11: Captura de pantalla de la herramienta visualizando la forma normal de Epirubicin

Capítulo 6

CONCLUSIONES Y LINEAS FUTURAS

6.1 Consecución de los objetivos

En líneas generales, se puede afirmar que se ha conseguido cumplir con los objetivos marcados para este trabajo de fin de grado. Se ha conseguido desarrollar una herramienta capaz de realizar un análisis de los conceptos médicos incluidos en la terminología médica SNOMED-CT. Dicha herramienta es accesible desde múltiples dispositivos gracias a las nuevas tecnologías utilizadas. También se ha creado un servicio REST para que la información obtenida pueda ser utilizada por herramientas de terceros.

La elaboración de esta herramienta ha necesitado un análisis profundo de la terminología incluida, SNOMED-CT así como un análisis de las tecnologías disponibles para realizar la herramienta. Estos objetivos se han cumplido en una fase temprana del desarrollo al ser necesarios para la elaboración de la herramienta. También se han realizado diferentes test de rendimiento, usabilidad y fiabilidad para asegurarnos de la utilidad real de la herramienta.

Una lista concreta de los objetivos se puede observar a continuación.

- **Investigación de los lenguajes médicos y sus mecanismos de composición actuales:** Este objetivo se consideró como un paso necesario a la hora de crear la herramienta y comprender la información utilizada a lo largo de este proyecto. Dado que se trataba de un objetivo primario y necesario para la realización del resto del proyecto, este fue uno de los primeros objetivos cum-

plidos satisfactoriamente. Gracias a este objetivo comprendimos la estructura de las terminologías médicas y como obtener la información posteriormente utilizada por la herramienta.

- **Investigación de las herramientas técnicas disponibles para el desarrollo de la herramienta:** Este objetivo se marcó para investigar las diferentes posibilidades técnicas para la implementación de la herramienta prestando especial atención al rendimiento y el número de usuarios que podrán acceder a la misma. Hemos podido estudiar diferentes tecnologías que han ayudado a cumplir este objetivo eligiendo definitivamente algunas de las últimas tecnologías disponibles asegurándonos así el soporte por parte de la comunidad tecnológica durante un ámplio periodo.

- **Especificación de requisitos:** Desarrollar la especificación de requisitos para la herramienta a partir de las necesidades esperadas de los usuarios. Consideramos diversos aspectos de los usuarios para escribir esta especificación pudiendo obtener un listado de los mismos capaces de cumplir las necesidades básicas de estos usuarios.

- **Implementación de la herramienta:** Este objetivo implica utilizar todo lo aprendido en los dos primeros objetivos y atendiendo a la especificación de requisitos del punto anterior, desarrollar la herramienta para el estudio automático de términos compuestos. Dada la preparación en los objetivos previos, este objetivo se ha podido cumplir obteniendo finalmente una herramienta que nos permite realizar un estudio profundo de los conceptos de la terminología médica SNOMED-CT.

- **Comprobación de los resultados:** Tras implementar la herramienta debíamos asegurarnos que los resultados cumplen con los requisitos especificados. También debemos comprobar la validez de estos resultados. Hemos podido realizar diversas pruebas de rendimiento a la herramienta obteniendo resultados satisfactorios y hemos podido comprobar la validez de los resultados comparándolos con otros visores de información que, si bien no ofrecen la misma información que la proporcionada por la herramienta, nos permiten obtener dicha información de forma manual.

6.2 Conclusiones

Tras realizar este TFG se ha conseguido aprender una gran variedad de tecnologías relacionadas con la web semántica dado que utilizamos esta tecnología para el acceso a las terminologías médicas. También se ha aprendido la estructura de diferentes terminologías y la extensión y el funcionamiento de las estudiadas en este proyecto. Gracias a mi trabajo en el grupo de informática biomédica de la facultad de informática de la UPM este conocimiento ha resultado de gran utilizad en mi trabajo.

También hemos conseguido aprender nuevas y diferentes tecnologías para presentar toda la información analizada a través del sistema de comunicación más ampliamente extendido en la actualidad, Internet. Además de diferentes tecnologías para transmitir esa información desde los servicios creados.

En definitiva, la realización de este proyecto ha permitido el aprendizaje de una variedad de tecnologías útiles para mi futuro profesional además de la satisfacción de la creación de una herramienta con una utilidad práctica y útil en diferentes ámbitos de la medicina y la informática biomédica.

Por otro lado se han encontrado desafíos relacionados con la variedad de conceptos con diferentes estructuras dentro de la terminología SNOMED-CT y la necesidad de un análisis profundo de una terminología con una enorme extensión como esta. También se han tratado algunos problemas en la presentación de algunos datos dada la enorme extensión de algunos de ellos. Por ejemplo, existen conceptos que están relacionados con más de 1.000 conceptos diferentes.

Pese a estos problemas consideramos que la herramienta cumple con eficacia su función y cuenta con posibilidades de ser una herramienta con utilidades prácticas en las distintas especialidades médicas.

6.3 Líneas futuras

La herramienta desarrollada en el presente TFG presenta grandes posibilidades de ampliación en el futuro. Algunas de estas posibilidades se explican aquí:

- **Inclusión de vocabularios:** Esta herramienta ha sido diseñada teniendo en mente la posibilidad de su ampliación a otros vocabularios, permitiendo buscar en una mayor cantidad de datos si incluimos vocabularios especializados.
- **Relación entre los vocabularios:** En el caso de que se incluyeran diferentes

vocabularios, se podrían realizar diccionarios de traducción entre ellos para facilitar la comprensión entre instituciones o individuos que utilicen vocabularios diferentes.

- **Marcado de términos:** La herramienta puede permitir el marcado de conceptos por los diferentes usuarios para facilitar el acceso a los mismos. Esto conllevaría tener un registro de los usuarios para almacenar sus búsquedas y sus registros.

- **Creación de términos postcoordinados:** Se puede ampliar la herramienta para ofrecer la posibilidad de elaborar términos postcoordinados a partir de los conceptos ya almacenados. Gracias a estos nuevos términos postcoordinados se podría ampliar un vocabulario en el cual no se encuentra el término necesario. A su vez si disponemos de diferentes vocabularios se podría considerar crear estos conceptos postcoordinados a partir de conceptos de diferentes terminologías indicando siempre la terminología de la que son origen los términos simples.

Como se puede observar, existe un amplio abanico de posibilidades para continuar y ampliar el desarrollo de la herramienta y ampliarla. En este TFG hemos implementado las funcionalidades necesarias para un funcionamiento básico de la herramienta.

Bibliografía

[1] Donnelly K, SNOMED-CT: The Advanced Terminology and Coding System for eHealth. Medical and care compunetics 3 2006: 279-290.

[2] Bodenreider O, Smith B, Kumar A, Burgun A, Investigating subsumption in SNOMED CT: An exploration into large description logic-based biomedical terminologies. Artif Intell Med, March 2007 39(3): 183-195.

[3] College of American Pathologists. SNOMED Clinical Terms Guide. Transforming Expressions to Normal Forms. August 2006. `www.cap.org/apps/docs/snomed/documents/transformations_to_normal_forms.pdf`

[4] Fp7-integrate.eu [homepage on the internet]. Driving excellence in integrative cancer research [updated 23 November 2012; cited 25 November 2012]. Available from: `http://www.fp7-integrate.eu/index.php/project`

[5] Paraiso S, Perez D, Alonso R, Claerhout B, Schepper K, Hennebert P, Lhaut J, Leeuwen J, Bucur A. Semantic interoperability solution for multicentric breast cancer trials at the INTEGRATE EU Project. In Proceedings of the 6th International conference in Health Informatics 2013.

[6] Eurecaproject.eu [homepage on the internet]. Enabling information re-Use by linking clinical REsearch and Care [updated 2012; cited April 2013]. Avalable from: `http://eurecaproject.eu/about`

[7] IHTSDO. [homepage on the internet]. International Health Terminology Standards Development Organisation. [updated March 2013; cited April 2013]. Available from: `http://www.ihtsdo.org/snomed-ct/`

[8] McDonald CJ, Huff SM, Suico JG, Hill G, Leavelle D, Aller R, Forrey A, Mercer K, DeMoor G, Hook J, Williams W, Case J, Maloney P. LOINC, a Universal Standard for Identifying Laboratory Observations: A 5-Year Update. Clinical Chemistry. 2003; 49 No. 4: 624-633.

[9] Seal RL, Gordon SM, Lush MJ, Wright MW, Bruford EA. Genenames.org: The HGNC resources in 2011. Nucleic Acids Research. 2011; 39: 514-519.

[10] Brown G Elliot, Wood L, Wood S. The Medical Dictionary for Regulatory Activities (MedDRA). Drug Safety, vol 20, Issue 2; 109-117.

[11] Beeler G, Case J, Curry J, Hueber A, Mckenzie L, Schadow G, Shakir AM. HL7 Reference Information Model. 2003.

[12] Tim Benson. Principles of Health Interoperability HL7 and SNOMED. London: Springer-Verlag; 2010.

[13] Cheetham E, H. Dolin R, Markwell D, Curry J, Gabriel D, Hausam R, Knight B, Rector A, Spackman K, Townend I. Using SNOMED CT in HL7 v3 Implementation Guide, Release 1.5. 2008.

[14] Berjon R, Leithead T, Doyle E, O'Connor E, Pfeiffer S. HTML5 A vocabulary and associated APIs for HTML and XHTML. 17 December 2012.`http://www.w3.org/TR/html5/`

[15] Gauchat Juan D. El gran libro de HTML5, CSS y Javascript. 1ª Edición Enero 2012.

[16] Etemad Elika J. Cascading Style Sheets (CSS). 12 May 2011. `http://www.w3.org/TR/CSS/`

[17] McFarland David S. Javascript and Jquery, the Missing Manual. O'Reilly 2nd Edition. 2012

[18] JQuery homepage `http://jquery.com/`

[19] MongoDB homepage `http://docs.mongodb.org/manual/`

[20] Twitter ,Inc. Bootstrap framework homepage. `http://twitter.github.io/bootstrap/`

[21] Armeli-Battana S. MVC Applies to Javascript. Developer.Press, Ed. 1. January 2013.

[22] Backbone Homepage. `http://backbonejs.org/`. Seen at May 2013.

[23] Underscore Homepage. `http://underscorejs.org/`. Seen at May 2013.

[24] Raphael.js Homepage. `http://raphaeljs.com/`. Seen at May 2013.

[25] Dahlström E, Dengler P, Grasso A, Lilley C, McCormack C, Schepers D, Watt J. Scalable Vector Graphics (SVG) 1.1 (Second Edition). 16 August 2011. http://www.w3.org/TR/SVG/.

[26] Joint.js Homepage. http://www.jointjs.com/. Seen at May 2013.

[27] Node.js Homepage. http://nodejs.org/. Seen at May 2013.

[28] Allamaraju S. RESTful Web Services Cookbook. Yahoo Press, Ed 1. 2010.

[29] JSON Specification. http://www.json.org/. Seen at May 2013.

[30] Express.js Homepage. http://expressjs.com/guide.html. Seen at May 2013.

[31] Hadley M, Sandoz P. JSR-000311 JAX-RS: The Java API for RESTful Web Services (Final Release). 2012. http://jcp.org/aboutJava/communityprocess/final/jsr311/

[32] Gosling J, Joy B, Steele G, Bracha G, Buckley A. The Java Language Specification, Java SE 7 Edition. 28 February 2013. http://docs.oracle.com/javase/specs/jls/se7/html/index.html

[33] Tomcat Documentation. http://tomcat.apache.org/tomcat-7.0-doc/. Seen at May 2013.

[34] Sesame Homepage. http://www.openrdf.org/about.jsp.SeenatMay2013.

[35] W3C OWL Working group. OWL 2 Web Ontology Language Document Overview (Second Edition). 11 December 2012. http://www.w3.org/TR/2012/REC-owl2-overview-20121211/

[36] Manola F, Miller E, McBride B. RDF Primer. 10 February 2004. http://www.w3.org/TR/rdf-primer/

[37] Brickley D, Guha R.V., McBride B. RDF Vocabulary Description Language 1.0: RDF Schema. 10 February 2004. http://www.w3.org/TR/rdf-schema/

[38] Prud'hommeaux E, Seaborne A. SPARQL Query Language for RDF. 15 January 2008. http://www.w3.org/TR/rdf-sparql-query/

[39] Hartig O, Zhao J. Provenance Vocabulary Core Ontology Specification. 14 March 2012. http://trdf.sourceforge.net/provenance/ns.html

[40] Miles A, Bechhofer S. SKOS Simple Knowledge Organization System Reference. 18 August 2009. http://www.w3.org/TR/2009/REC-skos-reference-20090818/

Apéndices

Apéndice A

Manual de instalación de la herramienta

A continuación detallaremos los pasos a seguir para poner en funcionamiento la herramienta en cualquier máquina servidora con acceso a internet.

1. Si no lo tienes instalado, instala el JDK (Java Development Kit).`http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html`

2. Instalar Tomcat en el puerto por defecto, 8080. Para ello se pueden seguir las instrucciones adjuntadas en estas direcciones según tu sistema operativo:

   - **Windows:** `http://goo.gl/n4qIM`
   - **Ubuntu:** `http://goo.gl/TVFXk`
   - **Mac OS X:** `http://goo.gl/ezizQ`

3. Ampliar la máxima memoria disponible para el proceso del Tomcat a un mínimo recomendado de 4096MB. Esto se puede realizar añadiendo la siguiente línea de configuración a la sección donde se definen las variables del archivo "catalina.bat" (Windows) o "catalina.sh" (Ubuntu, Mac OSX).

```
1 set JAVA_OPTS=-server -Xmx256m
```

4. Descargar Sesame. Sesame se puede descargar desde `http://www.openrdf.org/download.jsp`. La versión de Sesame utilizada durante la realización de este proyecto es la 2.7.0.

5. Tras descargar Sesame debemos copiar los archivos "openrdf-sesame.war" y "openrdf-workbench.war" de la carpeta "war" de la distribución a la carpeta

"webapps" dentro del directorio de instalación de Tomcat. También copiamos el archivo "SNOMEDSearcher.war" proporcionado con la herramienta en la carpeta "webapps" de Tomcat. Tras eso arrancamos Tomcat.

6. Dentro de la dirección `localhost:8080/openrdf-workbench` debemos crear un "New Repository" y seleccionamos como ID y Title "SNOMED" (sin comillas). Después pulsamos Next y Create.

7. Ahora seleccionamos Add a la izquierda de la página del paso anterior y presionamos el botón "Select File". Seleccionamos el archivo SNOMED.owl que acompaña a la herramienta y presionamos Upload. Esto llevará un tiempo y ocupará bastante memoria del sistema (alrededor de 3gb).

8. Cuando salga el resultado del paso anterior debemos instalar Node.js en la máquina. Node.js se puede obtener desde `http://nodejs.org/download/`.

9. Una vez instalado Node.js con una ventana de comandos o terminal nos dirigimos a la carpeta Frontend que acompaña a esta herramienta. Desde allí ejecutamos el comando "node index.js".

10. Si todo se realizó sin problemas podrás visualizar la aplicación ejecutándose en la dirección `http://localhost:8081`. El puerto de ejecución del servicio y la dirección del repositorio de Sesame se pueden cambiar en el fichero index.js.

Apéndice B

Consultas SPARQL para obtención de datos

Como hemos comentado, para la obtención de los datos de las terminologías almacenados en Sesame utilizamos el lenguaje de consultas para datos en RDF SPARQL. Las consultas utilizadas para obtener esos datos son:

- **Consulta SPARQL para obtención del label de un concepto**

```
SELECT ?label WHERE {
  CONCEPT_CODE rdfs:label ?label.
}
```

- **Consulta SPARQL para obtención del concepto asociado del term info de un concepto**

```
SELECT ?terminfo WHERE {
  CONCEPT_CODE prv:containedBy ?terminfo.
}
```

- **Consulta SPARQL para obtener los padres de un concepto**

```
SELECT ?parentsCode ?parentsLabel WHERE {
  CONCEPT_CODE rdfs:subClassOf ?parentsCode.
  ?parentsCode rdfs:label ?parentsLabel.
}
```

- **Consulta SPARQL para obtener los hijos de un concepto**

```
SELECT ?childrenCode ?childrenLabel WHERE {
  ?childrenCode rdfs:subClassOf CONCEPT_CODE;
          rdfs:label ?childrenLabel.
}
```

- **Consulta SPARQL para obtener las relaciones de un concepto**

```
SELECT ?relation ?relValue ?relationTitle ?relValueTitle WHERE{
  CONCEPT_CODE (rdfs:subClassOf|owl:equivalentClass)/owl:
     intersectionOf/rdf:rest* ?listMembers.
  OPTIONAL{
    {?listMembers rdf:first/owl:onProperty ?relation;
          rdf:first/owl:someValuesFrom ?relValue.
     ?relation rdfs:label ?relationTitle.
     ?relValue rdfs:label ?relValueTitle.} UNION
    {?listMembers rdf:first/owl:someValuesFrom/owl:onProperty ?
       relation;
          rdf:first/owl:someValuesFrom/owl:someValuesFrom ?
             relValue.
     ?relation rdfs:label ?relationTitle.
     ?relValue rdfs:label ?relValueTitle.} UNION
    {?listMembers rdf:first/owl:someValuesFrom/owl:
       intersectionOf/rdf:rest* ?aux.
     OPTIONAL{?aux rdf:first/owl:onProperty ?relation;
           rdf:first/owl:someValuesFrom ?relValue.
        ?relation rdfs:label ?relationTitle.
        ?relValue rdfs:label ?relValueTitle.}
    }
  }
  FILTER (BOUND(?relation)&&REGEX(STR(?relation),"SCT","i"))
}
```

- **Consulta SPARQL para obtener los conceptos relacionados con un concepto**

```
SELECT ?relation ?relValue ?relationTitle ?relValueTitle WHERE{
  ?n owl:someValuesFrom CONCEPT_CODE;
     owl:onProperty ?relation.
     ?relation rdfs:label ?relationTitle.
     ?s rdf:rest*/rdf:first?/owl:someValuesFrom* ?n.
     OPTIONAL{
      ?c owl:someValuesFrom*/owl:intersectionOf ?s.
      OPTIONAL{
    ?k rdf:rest*/rdf:first? ?c.
    OPTIONAL{
      ?relValue (owl:equivalentClass|rdfs:subClassOf)/owl:
         intersectionOf? ?k;
          rdfs:label ?relValueTitle.
       }
      }
   }
  FILTER(BOUND(?relValue))
}
```

- **Consulta SPARQL para obtener sugerencias de conceptos según su título**

```
SELECT (SAMPLE(?label) AS ?title) ?concept WHERE {
  {?concept rdfs:label ?label.} UNION
    {?concept skos:altLabel ?label.}
  FILTER(REGEX(STR(?label),"TEXT","i"))
} GROUP BY ?concept HAVING(BOUND(?concept))
LIMIT 10
```

- **Consulta SPARQL para obtener sugerencias de conceptos según su título filtrando por concepto del term info**

```
SELECT (SAMPLE(?label) AS ?title) ?concept WHERE {
  {?concept rdfs:label ?label.} UNION
    {?concept skos:altLabel ?label.}
    ?concept prv:containedBy TERMINFO_CONCEPT.
  FILTER(REGEX(STR(?label),"TEXT","i"))
} GROUP BY ?concept HAVING(BOUND(?concept))
LIMIT 10
```

- **Consulta SPARQL para obtener el código de un concepto dado su título**

```
SELECT ?concept WHERE {
  {?concept rdfs:label "TEXT"@en.} UNION
  {?concept skos:altLabel "TEXT"@en.}
}
```

Todas estas consultas utilizan los siguiente prefijos

```
PREFIX prv:<http://purl.org/net/provenance/ns#>
PREFIX :<http://www.ihtsdo.org/>
PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>
PREFIX hl7:<http://test.org#>
PREFIX loinc:<http://www.loinc.org/>
PREFIX owl:<http://www.w3.org/2002/07/owl#>
PREFIX xsd:<http://www.w3.org/2001/XMLSchema#>
PREFIX skos:<http://www.w3.org/2004/02/skos/core#>
PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
```

Este documento esta firmado por

	**Firmante**	CN=tfgm.fi.upm.es, OU=CCFI, O=Facultad de Informatica - UPM, C=ES
	Fecha/Hora	Fri Feb 14 18:52:52 CET 2014
	Emisor del Certificado	EMAILADDRESS=camanager@fi.upm.es, CN=CA Facultad de Informatica, O=Facultad de Informatica - UPM, C=ES
	Numero de Serie	630
	Metodo	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)